Java学习群

72030155

进群可以免费获取视频教程以及每日免费听老师讲课

Broadview
www.broadview.com.cn

互联网时代的软件革命
SaaS架构设计



叶伟 等著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



推荐序

1969年，互联网在美国诞生时，没有人会想到这项发明会改变整个世界。1971年，E-mail的诞生改变了人们相互的通信习惯。然而，在1990年，以HTML语言和HTTP协议为基础的WWW技术，让互联网真正成为了人们互动交流的信息工具。从此，互联网进入了飞速发展的阶段。当今的世界已被互联网完全打通，曾经阻碍信息沟通和交流的千沟万壑也早已被夷为平地。超高速和大容量的信息交换，使得整个人类社会的生产生活都发生了质的改变，各行各业也都在互联网浪潮的洗礼中经历着自身产业的变革。

在软件行业，互联网技术与软件技术的融合诞生了在线软件。经历过大起大落的磨难之后，在线软件最终在理论、技术和实践三个方面得到了升华，从而诞生了SaaS软件。Software-as-a-Service（软件即服务）的概念真正抓住了人们使用软件的根本目的，彻底将软件从产品变成了服务，从而改变了人们对软件的观念，实现了对软件本质的回归。

然而，受全球金融危机的影响，中小企业也遭遇了罕见的冬天，资金压力明显。而SaaS的低成本、按需使用和按需付费等优势，无疑给中小企业的信息化提供了最好的方案。同时，SaaS先天所具有的防盗版和大量互联网用户资源的优势，也给整个软件产业带来了前所未有的机遇，让业界看到了新的希望。

SaaS软件之所以受到业界重视，不仅仅是因为软件提供形式的转变，最重要的是SaaS为业界提供了一整套解决软件生产和消费问题的思想和方法。因此，SaaS不仅仅是软件行业的一次技术革新，更是催生了一个崭新的商业运营模式。这种创新商业模式的价值和作用无可估量，必将在互联网时代引发一场翻天覆地的软件革命！

伴随互联网发展而兴起的云计算技术，也正在从理论走向成熟。云计算提供的强大软硬件环境和基础服务，将逐渐成为支撑SaaS应用的基础设施。各个云计算平台所包含的大量具有自身特色的公共服务，为SaaS应用的开发提供了丰富的资源。而统一整合各个云计算平台的公共服务，也成为了SaaS服务集成平台（SIP）的发展目标。未来的SaaS应用将架构在由SIP整合的多个云计算平台之上。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

当前，软件和互联网行业已经呈现了三大发展趋势。第一，通过互联网使用软件的用户越来越多；第二，软件和服务的界限会越来越模糊；第三，软件和服务会越来越个性化。这些都促使Microsoft和Google等一大批全球知名企业，纷纷加入到软件与互联网相互融合的大潮中，SaaS市场的竞争亦愈演愈烈。Salesforce.com在美国的成功（2007年营收5亿美元）已经为业界树立了优秀的典范，但软件的未来更需要有一面指引前进方向的大旗。

与此同时，随着网络技术和软件技术的更新换代，打造SaaS软件所需的环境和技术都业已成熟。不过，开发SaaS软件的思想和方法与开发传统软件还是有很大的不同，这些不同点体现在设计思路、开发方法、应用架构以及系统优化等多方面。然而，整个软件界还缺乏SaaS开发的理论指导和专业经验，这无疑给SaaS产业的发展造成了阻碍。

时代的进步呼唤软件行业应该有一本SaaS方面的权威书籍，本书也就是在这样的背景下诞生的。它是中国第一本全面介绍SaaS理论和技术架构的专业书籍（全球第二本），系统地阐述了SaaS应用软件设计的方方面面。软件产业发展到今天，大量的传统软件等待改造，也有很多软件企业正想在SaaS行业大显身手，本书的推出无疑是雪中送炭。在互联网飞速发展的今天，相信本书能为广大的软件企业指明行业发展方向，提供宝贵的实践经验，并让我们满怀信心地迎接软件革命的到来！



阿里巴巴集团资深副总裁
阿里软件（上海）有限公司CEO
国家电子商务标准化总体组副组长

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

本书创作历程

2007年初，阿里软件公司成立，她的唯一使命就是发展中国的SaaS软件。于是，一大批国内顶尖的业界专家和技术领头人，纷纷加盟到阿里软件的SaaS大军中。凭借这群SaaS精英的智慧和劳动，阿里软件迅速崛起为国内最大的SaaS软件平台运营商。

阿里软件之所以能迅速发展，得益于其独特的SaaS平台商业模式。与一般的SaaS模式不同，阿里软件将阿里巴巴所拥有的大量电子商务资源对外开放，提供给第三方软件商开发增值应用。阿里软件互联平台所拥有的庞大高价值商业客户群，和极其丰富的电子商务API接口，以及大量引入的第三方服务，吸引了众多的软件厂商加盟阿里软件互联平台。

然而，在大量的传统软件商向SaaS转型的过程中，却遇到了难题。他们在SaaS理论、技术和经验方面，都缺乏系统的指导。这不仅仅是阿里软件互联平台的开发商们遇到的难题，也是整个中国传统软件企业向SaaS转型时面临的困惑。



身为阿里软件研发中心的技术总监，叶伟深深地陷入了思考。的确，尽管现在SaaS软件发展势头迅猛，但有关SaaS方面的技术资料实在太少。网上能搜到的，大都是新闻式的宣传信息，而真正能系统介绍SaaS理论和技术的使用资料几乎没有。而经过两年的实践，阿里软件无疑在SaaS方面积累了丰富的经验。作为阿里巴巴一贯的传统，以自身的发展来推动行业进步是阿里软件义不容辞的义务。该向业界输出技术和经验了！作为阿里软件权威的SaaS专家，叶伟深知自己的责任重大。

于是，叶伟找来了阿里软件首席架构师赵进，一起探讨如何帮助众多的第三方软件商完成向SaaS软件转型，以及如何向业界输出阿里软件的技术和经验等一系列问题。



赵进，为人朴实无华，行事成熟老练，总是面带微笑。凭借宏观的战略眼光和深厚的技术功底，年纪轻轻的他已坐上了阿里软件首席架构师的交椅。赵进的职业发展历程本身就可以写一部程序人生的传奇故事。他也是大家公认的SaaS技术权威专家，特别是在SaaS应用架构方面更是有独到的见解。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

经过叶伟和赵进的讨论，两位大侠一致认为，不能再用举办技术培训班或技术交流会等老办法来推广SaaS技术，而应该用更加系统和广泛的形式来向业界分享阿里软件的经验。写一本书！对，写一本书！写一本系统介绍SaaS理论思想和技术经验的书！这就是向业界推广SaaS软件的最好方法！

商议妥当之后，叶伟和赵进立即找来了阿里软件SaaS领域各方面的技术专家商议。大家一致认为写一本SaaS专业书籍的时机和条件都已成熟，而且这件事情意义非常重大，每位专家都很愿意奉献自己的技术和经验。于是，一个由叶伟牵头，赵进担任主编的写作小组迅速成立。除叶伟和赵进之外，参与编写的还有曾义、叶军、莫建祥、闻波、黄晓龙、龙良和李战，共9位SaaS各方面的技术专家。

经几天的构思，大家慢慢理出了整本书的写作大纲。然后，大家将各个部分进行了分工，每位专家负责自己最擅长部分的技术写作。为了增加全书的趣味性，赵进提出用一个有关SaaS的创业故事，来贯穿全书的内容。这一提议立即得到大家的一致认可，于是秉承阿里巴巴一贯的武侠文化传统，郭靖和杨康就成了本书中各种商业需求和实现技术的代言人，他们的创业历程也就是从传统软件逐步走向SaaS软件的过程。

在接下来的两三个月里，各位专家纷纷投入到充满激情的创作中去了。不过，写作过程确实不同于编写代码和技术文件，而且这本书是第一本SaaS权威书籍，大家也深怕写不好。于是，各位专家总是边写边交流沟通，互相学习，取长补短，同时也希望每个人的写作能保持一致的风格。整个写作过程也就在这种轻松愉快的气氛中推进。



李战，虽然是阿里软件最老的架构师，但却是个地地道道的老顽童，武功自然也是深不可测。虽然修炼SaaS的时间不长，却深得SaaS的精髓。由于李战的写作功底深厚，第一部分“SaaS商业模式”的编写也就顺理成章地落到他的头上，还有最后的结束语也是他的杰作。此外，他还承担了整本书的文字检查工作，一字一句地修订每一章节，对本书的贡献也是功不可没。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

赵进同学除了担任总编之外，还承担了第2、4、6章的编写工作。谁让他是首席架构师呢，这几章的内容编写非他莫属，能者多劳嘛。赵进所负责的章节部分也是最早完成的，也为其他几位作者写作树立了典范。当然，除了自己部分的编写外，他还细心指导其他章节的编写工作，给予大家莫大的帮助。本书能保质保量并按时完成，赵进应当是第一功臣。



黄晓龙，典型的技术专家，总给人一种稳重与实力并存的深刻印象。他在SaaS应用架构设计方面，也是大家公认的专家，而且实践经验非常丰富。在阿里软件核心的在线财务软件项目中，黄晓龙同学一肩挑起了主力架构师的任务，让这一项目成为阿里软件互联平台的杀手级SaaS应用。在本书的写作中，他负责完成了第3章的内容编写。其细腻的文字功底和通俗易懂的写作风格，也给大家树立了榜样。



龙良，国家认证系统分析师，阿里软件SaaS方面的又一重量级人物。他的重量不仅仅体现在体重方面，更重要的是其在SaaS的可配置与可定制性，以及SaaS安全性等方面都有很深的造诣。他现担任阿里软件“E网打进”产品的系统架构师，该产品是阿里软件互联平台的又一杀手级SaaS应用。第5章和第7章的内容自然也是他的势力范围，这两章内容又为整本书加上了重重的砝码。



闻波，生得一副娃娃脸，但却从眼镜后面那深邃的目光中，透露出务实稳健的成熟作风。闻波在客户端软件的架构设计方面，堪称阿里软件的泰山北斗。在阿里旺旺这款拥有9100万用户量的IM产品中，闻波担当了客户端产品的核心架构师。同时，他也是SaaS的离线应用架构方面的权威专家。第8章的内容就是他在离线应用方面多年的实践经验和心得总结，也是本书中的又一个重要章节。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



莫建祥，人称老莫，是阿里软件又一重量级牛人。11年大型系统开发经验，9100万用户量IM产品服务器的首席架构师，并在海量和分布式存储数据库集群方面的经验丰富。本书的第9、10、11和12章都是他贡献的精品之作，也是本书中份量最重的部分。



叶军，人称叶博，阿里巴巴的最优秀的博士之一。叶博从事SaaS研究多年，发表过的学术文章和论文众多，其论文经常被EI、SCI等国际权威检索收录。除理论方面的造诣之外，叶博的技术实力也是人人称道的，特别是在Open API方面更是成果斐然。第13章的内容就是叶博在Open API方面的精辟论述，也是叶博十年互联网生存的理论和实践的心得。



曾义，典型的玉面书生，性格温文尔雅。在他文静的外表下，掩盖了技术大牛的真正实力。他是阿里软件XPlatform核心架构以及SIP服务核心路由技术的掌门人，现任阿里软件研究院技术部经理。第14章对开放SaaS平台的论述就是曾义同学贡献的精品，都是开放SaaS平台方面不可多得的实用技术和宝贵经验。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

当各个部分的初稿基本完成之后，首先展开的是交叉阅读和评审，各部分负责人互相提出修改意见。在这个过程中，大家也发现了一些问题。比如，写作风格不太一致，编排形式不太一样，等等。于是，为了保持风格一致和前后呼应，大家相互参照，又开始了新一轮的修改过程。

经过近一个月的新一轮的修改，大家终于在2008年10月下旬完成了全书的第二稿。紧接着，更加紧张的评审工作开始了。于是，由叶伟亲自牵头，赵进主持，大家聚在一起每一章每一节地阅读和评审。

在评审的那段时间，华星科技大厦9楼北峰亭的灯光常常要亮到凌晨一两点钟。华星科技大厦9楼本来就是马云最早的创业办公室，有过战胜非典的光荣战绩，走出了阿里巴巴网络、淘宝网和支付宝等著名公司，阿里巴巴无数辉煌的历史似乎都与此地有关，而本书也是诞生在这样一个充满传奇的地方。



为了将这本历史性的著作写好，叶老大亲自操刀，一字一句地检查每一个内容环节。所有人都认真仔细地审视自己的章节，并对他人的章节细心审阅。尽管那段时间大家都有些疲倦，但却是精神饱满，信心十足。因为大家的心里只有一个想法，一定要将这本书写成精品，每一个技术环节都要能经得起业界的推敲。

经过近一个月的仔细评审，本书的第一版书稿终于完成。这是本书9位作者近半年辛勤付出的成果。当然，由于本书是第一本阐述SaaS理论和技术的书籍，而SaaS本身的理论和技术也在不断的完善和发展，书中难免存在一些片面性，自然也会存在一些不足的地方。我们也希望广大读者提出意见和建议，争取在后续版本中不断完善本书的内容。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。





与此同时，本书的创作引起了电子工业出版社博文视点公司的高度重视，并作为重点图书项目来对待。在整个图书编辑和设计过程中，博文视点的编辑们尽心尽责，精心处理图书的每一个细节。因此，本书的顺利出版也是与她们的辛勤付出分不开的。

最后，要真诚地感谢购买本书的读者们。有了你们的支持和鼓励，我们才会充满信心地完成本书的写作和出版工作。也希望大家对我们的工作提出实质性的批评和建议，我们一定会尽力改正，我们也一定会做得更好！

联系方式:

本书开辟了专门的博客，欢迎广大读者和我们交流和沟通。我们的博客地址是：

http://blog.csdn.net/AlisoftRD/category/476332.aspx

《互联网时代的软件革命》写作小组

2008年11月10日

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

前言

SaaS源于一种简单的思想：软件即服务！但SaaS的出现彻底改变了人们对软件的观念，使用软件的人从产品消费者转换为服务消费者，而开发软件的人从产品提供者转换为服务提供者。当然，SaaS带来的改变不仅仅体现在商业模式上，也带来了软件开发中的思路和技术的许多改变。这些都是整个软件界需要探讨和思考的问题。

当今的软件界已没有谁敢小觑SaaS软件，因为有一大批的如国外的Salesforce和国内的阿里软件这样的公司在这个领域取得了辉煌战绩，而Google和Microsoft以及国内的用友和金蝶等知名企业也纷纷加入SaaS大战之中。事实上，当前的软件行业正面临一场新的变革，导火索就是SaaS软件。

尽管有不少人对SaaS还缺乏了解，甚至对SaaS持怀疑态度；尽管SaaS本身也还存在这样或那样的缺陷，但SaaS软件发展的速度向世人证明了：软件行业从第二产业的生产制造业向第三产业的现代服务业进行大转换的潮流不可阻挡。

本书也正是在软件行业这样的大背景下推出的。这是国内第一本系统地介绍SaaS的理论和概念，以及SaaS软件的架构和实现技术的专业书籍。写作本书的目的主要是为了向软件界的朋友们分享有关理解和运用SaaS理论与技术的思想和经验。同时，本书的推出也希望能起到抛砖引玉的效果，让更多的朋友参与到SaaS理论和技术的探讨中，共同完善和发展SaaS的理论和技术，促进软件行业不断进步。

本书作者

本书是由阿里软件公司主持编写的，阿里软件是国内最知名的SaaS软件运营商之一。本书的作者都是阿里软件研究院的研究人员，在SaaS软件理论和技术方面有丰富经验。他们有些是SaaS理论权威专家，有些是SaaS软件的资深软件架构师，有些是阿里软件重大项目的技术带头人。他们分别是：

叶　伟 —— 阿里软件研发中心总监
赵　进 —— 阿里软件首席架构师
曾　义 —— 阿里软件研究院技术部经理
叶　军 —— 阿里软件研究院资深架构师
莫建祥 —— 阿里软件服务器首席架构师
闻　波 —— 阿里软件研究院资深架构师
黄晓龙 —— 阿里软件研发中心资深架构师
龙　良 —— 阿里软件研发中心资深架构师
李　战 —— 阿里软件研究院资深架构师

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

本书内容

本书共分15章，内容安排如下：

章名	主要内容	页码
第1章 SaaS的前世今生	介绍了SaaS的起源，阐述了SaaS的基本概念和理论。重点分析了SaaS软件给用户和软件商带来的好处和优势，当然也谈到了SaaS软件的缺陷及解决办法。还列举了一系列的SaaS成功案例，用事实论述SaaS商业模式的新成就	2～30
第2章 SaaS成熟度模型	阐述了SaaS软件的规模效应对软件架构设计的要求，而对规模效应的支持程度是衡量SaaS架构成熟度的重要特性。多用户环境下的高性能、可配置性和可伸缩性是成熟的SaaS应用应该具备的3个特性，而SaaS的成熟度模型正是根据应用对这3种特性的支持程度来确定的。本章也给出了在不同应用场景中正确选择SaaS成熟度模型的具体指导意见	32～47
第3章 构建Multi-Tenant应用 第4章 高性能的Multi-Tenant最佳实践	重点讲述了如何构建多租户的SaaS应用。从SaaS成熟度模型的Level1开始，一步一步讲解应用架构的改进，最终给出了Level4的高性能的多租户SaaS应用架构的最佳实践。构建高性能的多租户架构需要同时在数据库层、应用层和Web表现层3个方面进行改造和优化，本书也在这些方面给出了详细的解决方案	48～97
第5章 Multi-Tenant应用的可配置性 第6章 可伸缩的SaaS应用架构	分别介绍了如何构建可配置和可伸缩的多租户SaaS应用架构。SaaS应用的可配置主要包括数据可配置、功能可配置、界面可配置以及流程可配置等，可配置为SaaS应用的个性化设置提供了基础的支持。而对于可伸缩的架构能够保证在用户应用规模不断扩大的情况下，轻松增加相应的服务器等硬件设施，来保证SaaS应用的整体性能要求	98～147
第7章 SaaS系统安全	本章的内容重点是SaaS安全性。安全性保障是SaaS应用的重中之重，主要包括数据隔离安全、数据连接安全、数据保密、访问者认证、权限控制、监控和记录以及网络安全等多方面内容。本章也给出了解决这些问题的详细方案和技术手段。	148～169

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

本书内容

章名	主要内容	页码
第8章 离线应用	离线应用也是解决脱网使用SaaS应用的关键技术，同时也是提升SaaS应用性能的有效方法。本章内容涉及本地运行、本地存储、数据同步、冲突解决以及数据传输等各方面的技术，并给出了相关应用架构的设计方案	170～192
第9章 分布式文件存储 第10章 分布式数据库 第11章 分布式Cache 第12章 分布式计算	第9～12章是云计算基础设施的内容，也是本书的又一重点内容。这几章分别介绍了分布式文件、分布式数据库、分布式Cache和分布式计算等多种技术。其中详细讨论了GFS、HDFS和ADFS几种分布式存储方案，探讨了基于代理的分布式数据库架构，以及基于列的分布式数据库架构。在分布式缓存架构设计方面，提出了多层分布式Cache及其改进的架构设计。分布式计算主要探讨MapReduce的分布式计算框架原理，并以文件下载的分布式计算为例讲解分布式计算的各种架构和模型	194～258
第13章 Open API	主要讲解Open API技术，包括对Open API的商业价值分析、Open API相关产品、Open API实现技术、Open API托管平台及其服务集成平台实现方案。Open API的实现技术主要是SOAP和REST，在此基础上的Open API托管平台及服务集成平台可以向SaaS应用提供丰富和强大的服务接口资源	260～297
第14章 开放的SaaS平台	介绍了开放SaaS平台的概念。通过对PaaS几种商业模式的介绍，使读者对开放平台的真实特性更加清晰。对于SaaS应用来说，需要的不仅仅是狭义平台，更应该创造条件，完善价值链，吸引更多的角色参与到平台建设中，共同打造SaaS软件大市场和SaaS软件生态圈	298～323
第15章 结束语	探讨了SaaS未来发展趋势。从SaaS的平台化到SaaS的移动化，从SaaS产业链的形成到“服务即软件”思想的提出，SaaS软件的未来是丰富多彩的。此外，最后部分还探讨整个软件行业的发展趋势，并从信息产业大融合的高度分析了未来软件发展的产业终局	324～332

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

本书读者

由于本书所涵盖的内容较宽，适合阅读本书的读者群还是相当广泛的。当然本书主要还是为软件行业的开发人员准备的专业书籍，重点关注的是SaaS软件的架构设计与技术方案，而并非老少皆宜的初级读物。

如果你是软件或互联网公司的老板或高层管理人员，建议你应该看看本书的一些内容。你可以重点了解SaaS的理论知识和基本概念，SaaS软件的优势和缺点，以及SaaS所带来的稳健商业模式。当然，我们更建议你将本书推荐给你们的技术开发人员，他们往往能从技术上给予更多的评点。

如果你是IT企业的CTO或者技术负责人，本书将能为你提供较为全面的SaaS技术信息。特别是对于那些正打算转型到SaaS软件开发的技术公司，本书是不可多得的重要参考指南。书中所讲的很多内容都是业界关于SaaS的技术情报和行动计划，我们相信这些内容一定对你的技术决策有很大帮助。

如果你是软件公司的架构师，正在规划本企业SaaS产品的架构设计，本书就是专门为你量身定做的。SaaS软件的架构设计与传统软件有很大不同，对于从传统软件设计转过来的架构师们正需要这方面的参考和指导，而本书的重点就是讲述SaaS软件的架构设计的。

如果你是一名程序员，是时候关注SaaS软件的开发方法了。尽管对于基层程序员来说编写代码的工作没有多大改变，但了解整个SaaS软件的架构原理之后，对编写高质量的代码一定有很大的帮助。SaaS软件所需要的多用户、高并发和大容量特性，也要求程序员编写更高质量和更稳定的代码。

如果你是一位即将毕业或打算从事软件开发的新手，将能从本书中获知当前软件产业发展的最新动向和技术。其实，对于新手来说，先把握方向和目标是非常重要的，这样可以将有限的精力投到正确的学习和实践中，紧跟软件行业的发展潮流。我们可以肯定地说，未来软件的发展趋势一定是SaaS！

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

感 谢

对我来说，出版此书过程最大的痛苦在于交稿“时间”已定的情况下，如何保证“高质量”。此书是原创，很少有资料可以参考；并且每个人手头有大量的工作，很多时候只能用加班之后的加班来进行写作；而我这个头的内心中却有着完美情结，无法容忍逻辑不清、作图马虎、没有对比、分析不足、缺乏引导、拖沓冗长、用词晦涩、行文不畅、风格不一……衷心感谢我们写作小组的每一位成员，忍受住了我一次又一次的折磨，用无数个深夜与周末，献给各位读者这个还算过得去的作品。

——叶伟

感谢我们整个写作团队为此书付出的努力，是大家共同的努力，才能将一个简单的想法变成一本书。特别感谢叶伟对于全书严格、仔细的审查和修订，才能确保给我们的读者送上一本高质量的书籍。特别感谢李战对于全书所有文字仔细的修订，以及全书创作历程等的编写，为本书增色许多。感谢所有写作团队成员牺牲大量的休息时间，大家辛勤的付出是这本书最重要的基础。最后，还要特别感谢我的妻子陈晓云，在我写作这本书的过程中，她独自承担许多的责任，不仅毫无怨言，而且一直在鼎力支持我，让我能够更加投入到这本书的写作中，为读者朋友奉献更加精彩的内容。

赵进

本书凝聚了大家集体智慧和心血，充分体现了超越、信任、简易的公司价值观，表现了技术人员不断追求卓越的优良品质。在此，要感谢岑文初、吴峥涛等同事对于Open API、Cache和分布式存储方面的宝贵建议；也要感谢平台部门的其他同事，大家面对从未有过的挑战，积极乐观的面对，使开放平台的生态环境日益完善；最后要感谢写作组的全体同仁以及叶伟的悉心指导，你们的激情和对技术的执着，激励着我在今后的工作中不断前进。

——曾义

值此书籍定稿之际，衷心感谢尊敬的导师四川大学李志蜀教授。在攻读博士的四年多时间中，李老师给我的正确而深邃的指导使我能够深入研究网格计算和云计算相关技术。同时将我诚挚的感谢送给我新婚的妻子马妍妍，在我进行写作的时间承担了大量家庭琐事，给了我莫大的支持和帮助。

——叶军

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

感 谢

终于定稿了！第一次写书，真的感觉非常累，收获也很多。非常感谢我的妻子邵新阳，在写作期间给予我的支持和鼓励。同时也非常感谢阿里软件给了我一个宽松的环境，以及叶老大、李战等的辛苦修正。

——莫建祥

在本书撰写期间我母亲患有重病，衷心感谢我母亲对我的理解和支持。感谢我的父亲在我写作期间细心照顾着我的母亲，让我能有更多时间整理书稿。感谢叶老大、黄晓龙、李战的Review和修订。

——闻波

看到自己参与的著作即将出版，几个月来每个周末的苦思冥想和连续的挑灯夜战都化成了如释重负后的一股轻松感和成就感。其实写作是一种很好的总结和提炼过程，通过这次高强度的写作和碰撞，我们每个人对SaaS架构的理解都更进了一步。最后我将这本第一次参与完成的著作送给我聪明可爱的儿子——天天！

——黄晓龙

在此要特别感谢我的妻子刘丽贤、女儿龙心怡和我年迈的父母，他们为我多年的学习和工作提供了很多的支持和帮助。感谢阿里软件的各位同学，特别是叶伟、赵进和本次写书小组的各位同学，从你们那里我学到了很多；还有“E网打进”产品线的同学，感谢你们在本书写作期间为我分担了许多项目任务，谢谢你们的支持！

——龙良

感谢阿里软件给了我另一片新天地，在这里我玩得很开心。我要向我的儿子证明，老爸也是很能干的，老爸一直在努力，儿子也要加油哦。感谢叶老大、赵进、曾义、叶博、老莫、闻波、晓龙和龙良，从你们那里我学到了很多，我又长大了，哈哈……

——李战

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目　录

CONTENTS

SaaS 架构设计

第 1 篇　SaaS 商业模式

第 1 章　SaaS 的前世今生 2

1.1　什么是 SaaS 3

1.1.1　SaaS 的产生 3

1.1.2　SaaS 基本概念 6

1.1.3　SaaS 与 ASP 的异同 8

1.2　SaaS 软件的优势 9

1.2.1　给用户的优势 10

1.2.2　给软件商的优势 14

1.3　SaaS 的缺点及解决办法 16

1.3.1　依赖互联网 16

1.3.2　数据安全性 17

1.3.3　数据保密性 19

1.4　SaaS 成功案例 20

1.4.1　Salesforce.com 20

1.4.2　Google.com 22

1.4.3　Alisoft.com 24

1.4.4　其他 SaaS 案例 26

1.5　小结 29

第 2 篇　SaaS 应用架构

第 2 章　SaaS 成熟度模型 32

2.1　SaaS 的规模效应 33

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目　录

CONTENTS

互联网时代的软件革命

2.2　SaaS 成熟度模型分级 …… 36
2.2.1　Level 1：定制开发 …… 36
2.2.2　Level 2：可配置 …… 37
2.2.3　Level 3：高性能的多租户架构 …… 39
2.2.4　Level 4：可伸缩性的多租户架构 …… 41
2.3　如何选择合适的 SaaS 成熟度模型 …… 43
2.3.1　产品所面向的客户群的特征与需求 …… 43
2.3.2　产品的租户数量级别 …… 44
2.3.3　团队的开发能力与愿意付出的开发/改造成本 …… 45
2.4　SaaS 软件的成熟度模型的渐进步骤 …… 46
2.5　小结 …… 47
第 3 章　构建 Multi-Tenant 应用 …… 48
3.1　第一阶段：做项目 …… 49
3.1.1　发现商机 …… 49
3.1.2　“4+1”视图 …… 50
3.1.3　设备托管 …… 58
3.2　第二阶段：做产品 …… 59
3.2.1　设备共享 …… 59
3.2.2　创业之旅 …… 61
3.2.3　可配置化 …… 62
3.3　第三阶段：多租户 …… 63
3.3.1　成长的烦恼 …… 63
3.3.2　如何转化成 SaaS 多租户模式 …… 65

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目　录

CONTENTS

SaaS 架构设计

3.3.3　SaaS 多租户设计 …… 70
3.3.4　SaaS 多租户模式的威力 …… 72
3.4　小结 …… 73
第 4 章　高性能的 Multi-Tenant 最佳实践 …… 74
4.1　数据库层性能优化 …… 75
4.1.1　建立合适的索引 …… 76
4.1.2　消除大数据表连接 …… 77
4.1.3　避免复杂 SQL …… 78
4.2　应用层性能优化 …… 78
4.2.1　Cache …… 79
4.2.2　统计和报表计算 …… 82
4.2.3　基于 Tenant 的索引搜索 …… 84
4.2.4　异步操作 …… 85
4.3　Web 层性能优化 …… 87
4.3.1　Web 开发的性能优化策略 …… 88
4.3.2　Http 服务器的性能优化策略 …… 91
4.4　性能监控 …… 93
4.5　优化的效果 …… 95
4.6　小结 …… 95
第 5 章　Multi-Tenant 应用的可配置性 …… 98
5.1　数据可配置 …… 100
5.1.1　定制字段 …… 101
5.1.2　预分配字段 …… 102

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目 录

CONTENTS

互联网时代的软件革命

5.1.3 名称值对 …… 104
5.1.4 方案比较 …… 106
5.2 功能可配置 …… 107
5.2.1 原子功能划分 …… 108
5.2.2 功能包设计 …… 111
5.2.3 功能使用校验 …… 115
5.3 界面可配置 …… 116
5.3.1 系统菜单可配置 …… 116
5.3.2 页面元素可配置 …… 118
5.4 流程可配置 …… 119
5.5 配置元数据管理 …… 119
5.5.1 配置元数据 …… 120
5.5.2 租户配置数据 …… 123
5.5.3 配置元数据服务 …… 125
5.6 可配置系统运行 …… 125
5.6.1 系统菜单框架 …… 126
5.6.2 功能页面容器 …… 127
5.6.3 扩展数据引擎 …… 128
5.6.4 功能引擎 …… 128
5.7 小结 …… 128
第 6 章 可伸缩的 SaaS 应用架构 …… 130
6.1 伸缩性（Scalable）的概念 …… 132
6.2 应用服务器层的水平扩展 …… 133

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目　录

CONTENTS

SaaS 架构设计

6.2.1　基于 Session 复制的水平扩展方式 …… 134
6.2.2　基于 Session Sticky 的水平扩展方式 …… 137
6.2.3　基于 Cache 的集中式 Session 实现水平扩展 …… 138
6.2.4　三种水平扩展方式的比较 …… 139
6.3　数据库层的水平扩展 …… 139
6.3.1　数据库的垂直切分 …… 140
6.3.2　数据库的读/写分离技术 …… 141
6.3.3　数据库的水平切分 …… 142
6.3.4　三种数据库层的水平扩展方案对比 …… 145
6.4　小结 …… 146
第 7 章　SaaS 系统安全 …… 148
7.1　应用安全 …… 150
7.1.1　身份认证 …… 150
7.1.2　权限管理 …… 153
7.1.3　日志记录 …… 156
7.1.4　应用监控 …… 157
7.2　数据安全 …… 158
7.2.1　数据隔离 …… 158
7.2.2　数据库连接安全 …… 159
7.2.3　敏感数据加密 …… 160
7.2.4　数据量监控 …… 164
7.3　网络安全 …… 165
7.3.1　安全传输 …… 165
7.3.2　网络攻击防范 …… 167

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目 录

CONTENTS

互联网时代的软件革命

7.3.3 网络监控 …… 168
7.4 小结 …… 168
第 8 章 离线应用 …… 170
8.1 系统分析 …… 171
8.1.1 离线应用的 4 大核心问题 …… 172
8.1.2 离线应用架构 …… 173
8.1.3 相关技术背景 …… 175
8.2 本地使用 …… 176
8.2.1 网络状态检测 …… 176
8.2.2 离线页面支持 …… 178
8.3 本地存储 …… 178
8.3.1 本地数据库支持 …… 178
8.3.2 本地数据存储 …… 179
8.4 数据同步 …… 180
8.4.1 增量数据处理 …… 180
8.4.2 多用户冲突 …… 183
8.5 数据传输 …… 185
8.6 离线应用开发工具介绍 …… 186
8.6.1 本地数据库 …… 187
8.6.2 本地服务 …… 188
8.6.3 工作者线程池 …… 190
8.6.4 Gears 待发展的内容 …… 192
8.7 小结 …… 192

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目　录

CONTENTS

SaaS 架构设计

第 3 篇　SaaS 基础架构——云计算

第 9 章　分布式文件存储 ····· 194

9.1　大文件的分布式存储 ····· 195

9.1.1　GFS ····· 195

9.1.2　HDFS ····· 197

9.2　小文件的分布式存储 ····· 202

9.2.1　ADFS（Alibaba Distributed File System） ····· 203

9.2.2　小文件存储的另外方案介绍 ····· 208

9.3　小结 ····· 209

第 10 章　分布式数据库 ····· 210

10.1　基于列的结构化分布式数据库 ····· 211

10.1.1　HBase 数据库数据结构 ····· 211

10.1.2　HBase 系统架构 ····· 214

10.1.3　HBase 实践 ····· 215

10.2　基于代理的分布式数据库 ····· 217

10.3　基于列的分布式关系数据库 ····· 237

10.4　小结 ····· 241

第 11 章　分布式 Cache ····· 242

11.1　单层的分布式 Cache ····· 243

11.2　多层的分布式 Cache ····· 245

11.3　改进的多层分布式 Cache ····· 245

11.4　小结 ····· 246

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目 录

CONTENTS

互联网时代的软件革命

第 12 章 分布式计算 248

12.1 MapReduce 分布式计算框架 249

12.2 文件下载的分布式计算 252

12.2.1 中心索引方式 253

12.2.2 分布式的结构化方式 254

12.2.3 改进模型 256

12.3 小结 258

第 4 篇 开放的 SaaS 平台

第 13 章 Open API 260

13.1 Open API 的商业价值分析 263

13.1.1 Open API 的出现、起源与现状 264

13.1.2 Google Maps API 带来开放潮流 265

13.1.3 Mashup 在 Open API 的时代 266

13.1.4 Open API 的分类 268

13.1.5 Open API 的系统架构与运营 276

13.2 Open API 相关产品 279

13.3 Open API 的实现技术 282

13.3.1 远程接口技术 282

13.3.2 远程接口设计最佳实践 289

13.4 Open API 与 API 托管平台 290

13.5 服务集成平台设计与实现 293

13.6 小结 297

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目　录

CONTENTS

SaaS 架构设计

第 14 章　开放的 SaaS 平台 298

14.1　PaaS 299

14.1.1　插件模式 301

14.1.2　七巧板模式 303

14.1.3　云计算模式 304

14.1.4　服务托管模式 310

14.2　互联提升价值 311

14.2.1　拓展软件的能力：聚合互联网资源 313

14.2.2　应用互联的价值链 313

14.2.3　基础设施提供 316

14.2.4　互联需开放平台 316

14.3　软件互联平台 317

14.3.1　软件大市场 318

14.3.2　软件生态圈 319

14.4　小结 323

第 15 章　结束语 324

15.1　SaaS 发展趋势 325

15.1.1　SaaS 平台化 325

15.1.2　SaaS 产业链 326

15.1.3　SaaS 移动化 327

15.1.4　服务即软件 328

15.2　软件行业的未来 330

15.2.1　通过互联网使用软件的用户越来越多 330

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目　录

CONTENTS

互联网时代的软件革命

15.2.2　软件和服务的界限会越来越模糊 …… 330
15.2.3　软件和服务会越来越个性化 …… 331
15.2.4　软件产业大融合 …… 331
参考资料 …… 333

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第1篇

SaaS 商业模式

当前，在软件行业流行着一个新的名词“SaaS”。特别是在与互联网相关的应用领域内，其流行程度绝不亚于一首流行歌曲。

可“SaaS”到底是什么？

它又是如何产生的？

对我们有什么用？

对软件业又会带来什么样的革命？

下面的章节将针对这些问题进行阐述。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 1 章

SaaS 的前世今生

1.1 什么是 SaaS ····· 3
1.1.1 SaaS 的产生 ····· 3
1.1.2 SaaS 基本概念 ····· 6
1.1.3 SaaS 与 ASP 的异同 ····· 8
1.2 SaaS 软件的优势 ····· 9
1.2.1 给用户的优势 ····· 10
1.2.2 给软件商的优势 ····· 14
1.3 SaaS 的缺点及解决办法 ····· 16
1.3.1 依赖互联网 ····· 16
1.3.2 数据安全性 ····· 17
1.3.3 数据保密性 ····· 19
1.4 SaaS 成功案例 ····· 20
1.4.1 Salesforce.com ····· 20
1.4.2 Google.com ····· 22
1.4.3 Alisoft.com ····· 24
1.4.4 其他 SaaS 案例 ····· 26
1.5 小结 ····· 29

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

1.1 什么是 SaaS

1.1.1 SaaS 的产生

软件是什么?

关于这一问题，也许不同的人有不同的看法。也许大家都记得上软件专业课程时，老师都会首先提出这个问题。而最终一定会讲到一个令人震撼的答案：

软件是计算机的灵魂!

没错，自从计算机诞生的那一天起，软件就成了计算机不可缺少的东西。计算机离开软件就是一堆废铁，正如肉体离开灵魂也不过是行尸走肉。因此，软件是计算机的灵魂!

这是从我们这些计算机专业人士的角度来看软件是什么的。那么从计算机的使用者角度，也就是从用户的角度来看，软件是什么呢?



在计算机诞生的初期，计算机的使用者还没有软件的概念。因为，软件大多数是与机器融为一体的。用户在使用计算机的时候，他认为是计算机为他提供了相关的功能。至于软件，那是专业人士干的事情，与用户无关。因此，在那个年代，计算机就是软件!

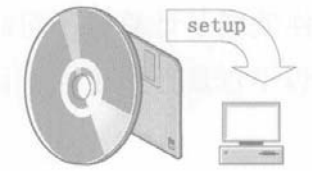


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

然而计算机是智能设备，智能设备的一个显著特征就是，可以通过改变内置的程序来使该设备具有不同的功能。因此，随着用户的计算机应用水平的不断提高，他们需要计算机去干越来越多的事情。而每次有新的事务需要用到计算机来处理时，不可能总是购买新机器。

于是，计算机的生产厂商也就开始向用户提供不同功能的程序。久而久之，这种单独提供程序的业务就分离出来，甚至有专门的人和公司来编写用户所需的程序。由此，诞生了软件产品，软件产品也逐渐支撑起了一个庞大的行业，就叫软件行业。

从此，用户不但需要花钱购买计算机，还需要花钱来购买所需要的软件，软件就成为了与其他商品类似的东西。这时，软件是一种产品！



软件是一种产品

随着软件工业的迅速发展，各种功能的软件层出不穷，用户可选择的软件产品也是琳琅满目。当今的信息社会，什么东西都可能用到计算机，各行各业几乎都会用到自己的专业软件产品。用户安装和使用到的软件是越来越多，而软件产品的功能也越来越强大。

为了支撑这些软件的运行，用户除了需要添置个人电脑、服务器和网络设备之外，往往还需要配备相关的系统维护人员。而用户的主营业务并非这些软硬件运行环境的维护，但却不得不维持这些投入和开销。因为软件已经成为企业工作中必需的东西，要保证软件正常运转，就得维持一个稳定的软件运行环境。

一些软件开发商看到了商机，提出了一种新的软件应用模式ASP。ASP是Application Service Provider 的缩写，意思是“应用服务提供商”。这种模式就是将用户需要的软件统一部署在应用提供商的软硬件环境中，软件运行所需的人力、物力资源都由提供商维持，而用户只需在自己的办公室使用这些软件即可。

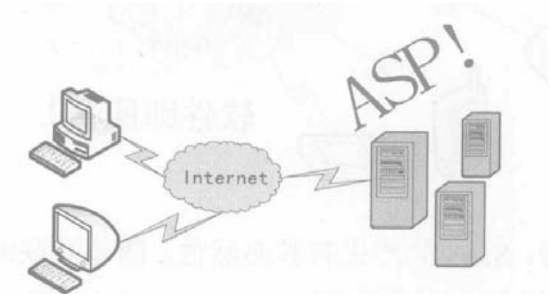
用户使用软件时，只需连到应用提供商的服务器，就可以处理自己的日常业务。同

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

时，业务中所有的数据资料也都存储在应用提供商的服务器中。当然，这需要互联网的支持，而且 ASP 软件大都是用浏览器作为用户启动软件的入口的。

ASP 模式就是将软件统一托管到应用服务提供商的服务器中。这时，软件是一种被托管的应用。



软件是被托管的应用

其实，ASP 已经有了为用户提供服务的初步想法，只是关注的重点在于提供软硬件环境这样的服务，而不是软件本身。然而，从用户的角度看，用户不仅仅关心软件的低成本地稳定运行，更关心的是软件能为自己做些什么实质性的工作。也就是说，用户更关心软件本身能为自己提供什么样的服务。

许多用户在使用了 ASP 模式的软件之后，发觉这种模式仅仅是改变了使用软件的形式。而且由于当时的网络带宽和软件技术上的限制，ASP 软件的用户体验也不是很好，根本无法与传统的桌面软件相比。同时，用户普遍都还无法接受将自己的软件和数据交由第三方来托管的方式。因此，ASP 维持了几年之后并没有取得想象中的成功，也就渐渐销声匿迹了。

令人欣慰的是，ASP 也为软件开发商们留下了宝贵的经验和教训，而这些经验和教训也让软件商不断地反思。其实，用户需要的并不是软件的运行环境的改善，用户使用软件的目的就是为了能让软件帮自己干活。从这个思路出发，说明用户实际上需要的是服务。软件不应该是产品，也不仅仅是提供应用托管，软件应该服务化。于是，软件开发商在多年的反思和总结基础之上终于又喊出新的口号：软件即服务！

软件即服务就是 SaaS，SaaS 的目标是软件的彻底服务化！SaaS 完全抛弃了软件产品的概念，并继承了 ASP 的形式，但并不拘泥于 ASP 形式，而将重点放在软件本身的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

服务能力上。



从软件的演化历史来看，SaaS 的产生有其必然性。随着互联网的飞速发展，软件市场需求的日益增长，软件生产力的大幅提升，以及用户对软件核心价值观的转变，必然导致软件的服务化趋势，从而催生 SaaS 商业模式。因此，SaaS 是在各种条件和时机成熟之后的必然产物，它的诞生也必将引发软件产业的一场翻天覆地的革命。

1.1.2 SaaS 基本概念

SaaS 是英文 Software as a Service 的缩写，中文的意思就是：软件即服务。

SaaS 的核心概念是软件的服务化。即，SaaS 将软件看作是一种服务，而不是一种商品。SaaS 的基本理论认为，软件的实质是服务。用户对软件的需求实际上是对应用服务的需求，而用户使用软件实际上是在消费应用服务。软件的用户是服务的需求者和消费者，而软件的提供商是服务的提供者和生产者。

从现代经济学的供需理论看，任何一种商业模式的建立，都需要一种供需关系的体系来支撑。在传统软件产业的各种商业模式中，大都是将软件当作一种商品。比如，定制项目软件、企业管理软件、个人消费软件以及共享软件，都是以开发、生产和销售软件商品为主的商业模式。这些软件商业模式与生产和制造业的商业模式一样，都将供需关系建立在对商品的供需这一基础之上。

然而，软件却是和一般商品不同的东西。首先，软件只有研发成本，没有生产成本。其次，软件是无形商品，没有损耗的概念。最重要的是，软件是需要持续地服务、

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

维护和升级，才能正常发挥其内在价值的。正是由于软件的这些特殊性，使得软件并不适合当作商品来销售。而传统软件商业模式中的种种弊端和问题，都是源于软件的这些特殊性。

事实上，用户使用软件的目的是希望软件能帮助其完成某方面的工作。因此，软件的用户关心的是软件的功能和处理能力，以及软件是否使用方便和运行稳定。至于软件的形式或形态，并不是软件用户关心的重点。而且对于软件来说，离开了软件供应商的支持和维护，很难能真正用起来。即使能用起来，离开软件供应商的服务，也很难用得好。因此，不管从用户使用软件的目的来看，还是从软件使用的过程来看，软件都是一种服务而不是商品。

SaaS 明确地将软件定义为服务，将供需双方对软件商品的需求定义为对软件服务的需求，从而将软件商业模式从商品供需模式转换为服务供需模式。这样，SaaS 软件的商业模式就脱离了传统生产制造业的那种简单模式，成为现代服务业中的一种特有的商业模式。从软件的供需关系上来看，将软件的供需关系定位为服务的供需关系，比将其定位为商品的供需关系来说，显得更加清晰，更加符合软件本身的特点，也更符合经济学的基本规律。



SaaS 商业模式理论的提出，将软件业从基于生产制造的第二产业，转变为基于服务的第三产业。这一转变是革命性的，彻底颠覆了传统软件业固有的经营模式。人类社会经过工业社会发展到信息社会，相应的经济活动也从以制造为中心转向以服务为中心，这是社会经济发展的大的趋势。软件行业也面临同样的问题，而 SaaS 软件的出现正好

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

符合了行业发展的潮流，也必将引起软件行业的一次巨大变革。

1.1.3 SaaS 与 ASP 的异同

但真要说 SaaS 和 ASP 到底有啥区别，还真的比较难。因为，SaaS 和 ASP 的形式太像了。有人说 SaaS 和 ASP 就是换汤不换药，甚至怀疑是故意发明的新名词来进行炒作的。下面我们就来简单说说 SaaS 和 ASP 的区别及联系。

- **看问题的角度不同**

ASP 是站在软件开发商的角度看问题的。ASP 是希望用一种新的形式向用户提供软件，而往往忽略了软件本身的价值。这与用户所关注的主要内容并不相符，因而无法引起用户的共鸣，也就很难被用户接纳。

SaaS 是站在用户的角度看问题的，考虑的是用户需要什么的问题。这样就搞清了提供软件的本质是为了提供服务，这与用户的实际要求是一致的，因而能被广大用户接受。立场决定观点，也决定了做事的方式方法。正是因为 SaaS 与 ASP 的立场不同，才决定了 SaaS 不会像 ASP 那样走向失败，而是为用户做得更多，最终走向成功。

- **关注的目标不同**

ASP 发展的初期，软件开发商只是为了解决用户软硬件环境的持续维护问题，因此提出应用服务的统一托管模式。所以，ASP 关注的是如何为用户提供一种新的软件应用模式，方便用户更容易地使用软件，并能降低软件的维护成本。

在 ASP 发展的后期，软件开发商才意识到 ASP 软件本身的用户体验很差，软件技术上还不够成熟。因此，开始将提高软件的可用性和易用性作为目标。但这些都是补救措施，却无法挽救 ASP 模式的败势。

而 SaaS 一开始就将开发的重点放在软件本身的功能上。SaaS 关注的是软件是否能为用户提供有效的服务，而并不将重点放在提供服务的形式上。因此，SaaS 软件大都有很实用的功能，有较好的用户体验。用户感觉 SaaS 软件和传统软件没有什么不同，甚

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

至功能还会比以前的软件有所提高。特别是某些与网络相关的应用，如电子商务服务和即时通信服务等，能让用户充分享受互联网带来的莫大好处。

随着互联网软件开发技术的不断进步，SaaS 软件将能为用户提供更多更强大的功能服务，这些功能性的服务将永远是 SaaS 关注的重点。至于提供服务的形式也将会越来越多，原来的 ASP 形式也将继续发展，传统的 C/S 结构的软件也将引入到 SaaS 模式中，而 RIA 形式的 SaaS 软件也会越来越多。

■ SaaS 与 ASP 的联系

应该说，SaaS 软件是在继承了 ASP 的已有经验，并用新的“软件即服务”的理论指导下，发展起来的新型软件商业模式。因此，SaaS 与 ASP 之间的关系实际上是形似而神不似。正是 SaaS 和 ASP 之间的这种貌合神离的关系，让很多人难以分清 SaaS 和 ASP。从 ASP 到 SaaS 可以用一句话概括，那就是“无可奈何花落去，似曾相识燕归来”！

不过，话又说回来，分清 SaaS 和 ASP 就真的那么重要吗？其实重要的并不是形式，而是由此给我们的工作和生活带来的影响和变革。在这一点上，SaaS 完全跳出了 ASP 的局限性，以一个更加通用和广泛的理论来指导软件行业的发展。

SaaS 是属于云计算这个更大的概念中的具体应用。尽管“云计算”这个概念正如它的名称一样还让人云里雾里的，但 SaaS 的概念却非常清晰和具体。SaaS 的理论已有大量的现实 SaaS 应用来支撑，这些现实的 SaaS 应用已经逐渐取得了商业上的成功。越来越多的软件厂商投入到 SaaS 软件的开发大潮中，追逐着朵朵浪花，奔向云计算的那一片宽阔的海洋。

SaaS 的前景是美好的，也必将带来软件行业的一次大变革。相信在互联网的这一片蓝天中，SaaS 一定是最美丽的那朵白云……

1.2 SaaS 软件的优势

SaaS 软件能得到强有力的发展，除了 SaaS 软件开发的推动之外，关键还在于 SaaS

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

软件本身的优势。就是 SaaS 这些自身的先天优势，才使得 SaaS 软件在与传统软件的竞争中能够攻城略地，甚至逐步夺取传统的软件统治的天下。

那么，SaaS 软件到底有什么样的优势呢?

1.2.1 给用户的优势

先从用户的方面来看。

■ **拿来即用**

传统软件需要首先向软件商购买软件，然后在自己的电脑或网络服务器上进行安装和部署。在这个过程中，也许还有各种较为复杂的配置，特别是对于一些稍大一点儿的企业管理软件，还往往需要专业人士的参与才能完成整个安装和部署的过程。

SaaS 软件却是可以拿来即用的。因为 SaaS 软件的安装和部署都是软件商的事情，用户根本无须关心软件安装的问题。用户需要使用软件时，只需在网上找到所需要的软件，点击即可开始使用了。无须下载，没有安装，没有部署，更不需要配置。

对于用户来说，使用 SaaS 软件是很简单的事情。只要能连上互联网就可以开始使用 SaaS 软件，根本没有传统软件那种复杂的安装和配置过程。SaaS 软件上线之前就已经在软件供应商的网络服务器中安装好了，软件的安装和配置都与用户无关。即使在初次使用 SaaS 软件时可能会有些初始化过程，但也大都是通过默认处理或自动处理来完成的。

■ **无须维护**

传统软件的全部运行环境都是由用户自己负责的，因此软件的运行维护也多半要由用户自己来解决。一般来说，企业都会有一到多位自己的计算机维护人员，他们负责和保障整个企业的各种软件正常运行。这些维护人员还会在软件公司的维护工程师的指导下，完成指定软件的问题处理和系统维护等工作。

而 SaaS 软件的情况就简单多了。SaaS 软件的主要运行环境绝大部分都在服务器端，

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在软件商的机房内。因此，大部分的系统维护工作都是由软件商承担的，并有经验丰富的专职工程师负责实施。

而用户前端的软件运行环境大多都是通用的浏览器，或其他公共工具软件（如即时通信软件），这些软件都是非常稳定的，很少出问题。并且，用户长期使用这些公共软件，本身已是经验丰富，即使有小问题往往都能自己解决。

因此，对于用户来说，根本无须关心 SaaS 软件的维护和升级问题。当用户在使用 SaaS 软件中遇到问题时，只需通知软件商的专业维护人员去处理，无须自己花费精力和想尽办法去解决。而且软件商的专业维护人员定位问题更准，解决问题更快。

在 SaaS 软件升级方面，用户的感觉也是很平滑的。当 SaaS 软件商改进了现有应用，或开发了新功能时，更新和升级都是在后台完成的，根本无须惊动使用软件的用户。也许用户在第二天进入 SaaS 软件时，会突然发现软件的一些新变化，看到了新的功能，这才知道软件有过升级换代。

SaaS 软件无须用户维护的特点，也是 SaaS 软件受到广大用户青睐的重要原因之一。

■ 按需使用

传统软件买了就买了，只能多买而无法退货。用户购买软件必须估计到软件的最大使用情况，否则在业务繁忙时，软件就无法支撑企业的业务处理。但当企业用不到那么多软件时，这些多余的软件也就成了闲置的资产，既不能退，又无法给企业带来效益。

因为，SaaS 的服务收费形式有多种，有的可以按使用次数收费，如：重要资料查询等服务；有的可以按使用人数和时间收费，如：各种管理软件；有的又是按处理的数据量来收费，如：短信和彩信处理服务。

所以，SaaS 软件就是可按需使用的服务。企业的人少时就少订购一些服务，规模扩大时就多订购一些服务，人员缩编时又能相应减少订购服务。业务繁忙时多用些服务，在淡季又可以少使用一些服务。甚至企业在歇业时也可以完全停止 SaaS 软件的使用，而过些年重新开业时又接着使用，原来的数据资料都在。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

SaaS 软件的服务就像水龙头里的水、电网里的电，你需要时就拧开龙头，打开开关，用多用少完全自己把握。这就是 SaaS 软件服务的按需使用！

■ 随处可用

传统软件的数据资料大都存在一个固定地方，与物理上的地理位置相关。数据或者在办公室的电脑中，或者在企业局域网的服务器中。而每当用户出差或休假中想要处理业务时，就没有办法了。

而 SaaS 软件的数据资料都是存在互联网中的服务器上，它的存储位置是虚拟的，甚至用户都不知道存放数据的具体物理位置。但是，只要在有网络的地方，用户连上互联网就可以使用 SaaS 软件，开始处理自己的文档和业务。

当今世界，除了极其偏远的地区，网络真的是无处不在。办公室、家里、网吧、宾馆、酒店，以及许多公共场所，都有网络接入口。有有线的，也有无线的；有收费的，也有免费的。一些城市还由政府出资，建造公共的免费无线网，打造网络城市。

因为网络无处不在，才使得 SaaS 软件随处可用。这使得 SaaS 软件模式在未来的网络社会中，有更强的生命力和竞争力。

■ 风险减小

在传统软件那种产品销售模式中，用户要想引入一套软件是一件很困难的事情，困难在于对该软件的应用评估和决策方面。的确，一套软件在没有深入用过以前是很难评价其好坏的。特别是一些企业管理软件，简单地试用是无法了解整个软件的情况的。

但软件产品必须购买之后才能合法使用，而购买软件往往需要一次性支付很大一笔金钱。如果购买的软件在使用一段时间之后，发现根本满足不了要求，或者不适合企业的管理模式时，将给企业带来巨大损失。因此，购买软件产品存在很大的风险。

SaaS 软件提供的是服务而不是产品，因此并不要求用户一次性支付很大一笔钱。用户只需支付很少一点费用就可以开始使用 SaaS 软件。因此，用户就可以以很少的一点投入开始软件的尝试，风险非常小。如果发现该软件并不满足要求或不适合本公司的管

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

理模式，也就算交了一点点学费，企业是完全可以承受的。这比起传统软件买断产品的方式来说，风险小多了。而且，有的 SaaS 软件还提供一个免费的试用期，这就更是将用户的风险降到零。

■ 成本降低

传统软件大都是按产品和服务分别计费的，有的软件还要收取升级费。用户第一次购买软件时需要支出很大的一笔费用，以后每年都会有一笔固定的维护和服务费用，遇到升级时可能还会有升级费用。

除看得见的成本之外，传统软件还需要自己维持一套设备和人员来保证软件的正常运行。虽然，这些支出被算在管理成本或办公成本当中，但却是因为使用软件而产生的成本费用。这也是企业在引进一套软件时不得不考虑的问题。

SaaS 软件的成本计算非常简单，就是按服务多少来收费。SaaS 软件是服务，用户购买的不是产品而是服务。因此，SaaS 软件只收取服务费，除此之外没有其他费用。此外，SaaS 软件也无须用户自己维持一套人马来保证软件的正常运行。软件的维护是软件商的服务内容之一，都已经算在服务费中了。因而，用户也能节约一大笔隐性的软件运营成本。

低成本地使用 SaaS 软件是吸引中小型企业的最好方法，SaaS 软件也因此受到众多中小型企业的追捧。

■ 先天防毒

计算机病毒无时无刻不在威胁用户的信息安全。有一定计算机信息安全防护经验的大企业，也都是在与计算机病毒的不懈斗争中成长起来的。而对于广大中小企业来说，却很难跳出计算机病毒的泥潭。它们缺乏专业的反病毒支持，即使购买了反病毒软件，也往往不会正确地使用。

对于传统软件来说，企业的计算机一旦感染病毒，往往带给企业的是一场灾难。特别是反病毒软件失效时，大量用户不得不经常重新安装系统，一个一个地安装应用软件，再一个一个地恢复数据。往往一不小心又染上了病毒，不得不重新来过。所有这一切都将会消耗企业大量的人力、物力，甚至影响企业正常的运转。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

而对于 SaaS 软件来说，软件本身是安装在 SaaS 软件商的网络服务器中的。SaaS 软件商都会有专业的安全保障措施，也对计算机病毒有严密的防范机制。一般来说，计算机病毒要想侵入 SaaS 软件服务器，几乎不可能。SaaS 软件的服务器端是非常安全可靠的。

SaaS 软件的客户端一般都是浏览器或其他工具软件。即使客户端的系统被病毒感染，只需重装系统和浏览器等工具软件即可，无须一一安装和恢复应用软件。在客户端中毒情况不严重的情况下，用户甚至可以带毒工作，等有空了再清理客户端的病毒。

客户端的病毒无论如何都很难传播到服务器，因为客户端和服务器主要是协议化的数据往来，病毒很难插进来。即使客户端可能上传带毒文件等，也会被服务器的病毒监控工具捕获并屏蔽。因此，SaaS 软件具有先天的防病毒特性。

1.2.2 给软件商的优势

那么，从软件开发商的角度看，SaaS 又有什么优势呢?

- **节省销售成本**

软件开发商在推广和销售传统的软件产品时，需要投入巨大的营销成本。特别是在销售方面，用户往往会仔细考察软件的各项功能，很难下决定。软件开发商不得不反复地给用户讲解和展示软件的功能，更多地与用户进行沟通交流，想尽各种办法说服用户购买软件。这往往在无形中增加了许多销售成本。

但在推广和销售 SaaS 软件中，软件商可以将重点资源放到市场推广方面。只要市场方面的工作做得好，在销售环节中很容易让客户开始使用软件，因为 SaaS 软件并不需要用户一次交付很大一笔费用。只要软件的功能做得好，用户自然会长期订购软件商的服务。

因此，SaaS 软件的营销主要在市场，销售是顺理成章的事情。这就为软件商节约了一大笔销售费用，节省的费用可以用于其他更有价值的支出中。

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

■ 节省维护成本

传统软件产品的维护是一对多的。也就是说，软件商的维护人员必须同时维护多套软件产品。维护人员不得不面对各种各样的用户现场环境，什么样的用户环境都有，常常会出现千奇百怪的问题。对于许多严重的问题，维护人员还不得不跑到用户的现场环境去解决。这零零总总的情形，都极大地加重了软件维护人员的负担。

更为糟糕的是，维护人员不但要维护不同的软件运行环境，甚至还需要维护同一软件的不同版本。这些复杂的情况往往使得维护人员疲于奔命，既消耗了大量的精力，维护的质量也不见得很高。这些都造成了软件维护成本居高不下的局面。

SaaS 软件的维护是一对一的。因为 SaaS 软件只部署在软件商的服务器中，只有一套软硬件运行环境，因此也只需维护一套软件的运行。同时，SaaS 软件维护人员对自己公司的软硬件环境非常熟悉，维护起来也会得心应手。

当用户遇到问题并通知软件商时，SaaS 软件的维护人员常常能在第一时间定位问题的具体位置，并迅速帮助用户解决问题。只维护一套运行软件的特点，充分保证了 SaaS 软件对客户服务响应的快捷和高效。

此外，SaaS 软件的维护还有一个优势就是：几乎不用到客户的现场就能搞定一切。因此，维护人员不用到处奔波，真可谓是“运筹帷幄，决胜千里”啊。这一切都使得 SaaS 软件在提高服务质量的同时，维护成本也大大降低。

■ 稳健的经营模式

在传统软件的营销中，软件公司的主要收入来源于用户第一次支付的货款。软件产品销售的收入占了整个收入的 80～90%的分量，而其他的服务费和维护费收入充其量占整个收入的 10%～20%，甚至更低。软件公司在完成对一家客户的销售之后，又不得不寻找新的客户继续推销产品，以此维持公司的生存和发展。因此，软件公司总是将重点业务放到软件的销售之上，软件公司的销售部门也是压力最大的部门。业务员成天忙忙碌碌地东奔西跑，电话、传真应接不暇，目的只有一个：签单。

将重点工作放到新客户身上，这是传统软件营销模式的普遍现象。当销售完成之后，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

由于服务和维护并不能给软件公司带来多大收入，就成了履行承诺的后续工作，很难让人有工作的激情。这就常常导致了客户服务和维护工作质量的下降，还常常收到客户投诉或抱怨。这一切并非人的问题，而是经营模式出了问题。

软件产品不同于其他有形产品，软件的客户服务和维护工作远远多于其他有形产品。或者说，软件的服务和维护才是软件产品的一切。而“软件即服务”正是完全将软件当作了服务，因此对软件的经营模式也就从销售模式转变为服务模式。

SaaS 软件的营销重点是市场，销售卖的是服务，比较容易展开。市场工作上去了，用户量也就会自然而然地提高。用户一旦开始使用 SaaS 软件的服务，只要软件符合用户的需求，用户用得满意，也自然会持续使用软件公司的服务。这就使得 SaaS 软件的用户群成了一种可以积累的资源，而且这些客户资源具有相对的稳定性，并且 SaaS 软件的销售收入也是很平稳及可以预测的。

因此，一旦 SaaS 软件开始营销，就能给软件企业带来持续的和稳定的，并可以预测的销售收入。这就为软件企业提供了一套平稳的运营机制，为企业构建了一套稳健的经营模式。

有了这种稳健的经营模式，软件企业就能将更多的精力放到产品的研发和推广中，促进企业不断提高客户服务质量和系统维护的水平。从而在软件企业与客户之间形成良性的循环，互惠互利地协作，真正能够达成软件开发商和软件使用者的双赢目标。

1.3 SaaS 的缺点及解决办法

任何事物都不是完美的，SaaS 软件也一样。那么，SaaS 软件又有哪些问题和缺点呢？针对这些缺点和问题，我们又该来如何化解呢？

1.3.1 依赖互联网

首先，SaaS 的第一个缺点就是需要互联网，互联网是 SaaS 软件赖以生存的根基。

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

因为 SaaS 软件是安装和部署在互联网中的服务器上的，用户需要通过互联网连上 SaaS 软件的服务器，才能正常使用 SaaS 软件的服务。离开互联网，SaaS 软件将无法使用，用户什么也干不了。

而传统软件是安装和部署在自己的个人电脑或者企业自身的服务器中的，互联网只是用来通讯的，没有互联网软件照样可以运行和使用。当然，在现在的传统软件中，有些功能也需要互联网的支持，比如数据传递和通讯等，但这些都不是传统软件的主要部分。当互联网中断时，只会对软件的使用造成有限的影响，而软件的主要功能仍然能继续使用。不依赖互联网的特点，也是传统软件的优势。

不过，当今社会，人们的工作和生活越来越依赖互联网。办公室的人每天都会收发电子邮件，使用即时通讯软件，浏览网络新闻，等等。其实，互联网已经成为人类生产和生活的一项基础设施，可以夸张地说，它的可靠性和稳定性直接关系到国计民生。

2006 年台湾海峡的一次强烈地震，造成亚洲通往美国的海底光缆大面积破坏。海底光缆是各大洲之间通讯的主干线，一旦被破坏将严重影响全球的通讯网络。一时间，全球网络出现了非常严重的堵塞，特别是亚洲各国的通讯用户。许多电子邮件无法处理，国际电话普遍无法打通，给人们的生产和生活造成严重的影响，甚至一些公司的业务出现混乱。

因此，人们已经越来越清楚地认识到，互联网是信息时代社会必不可少的基础设施。它就像电气化时代的电网一样，支撑着信息时代整个社会的正常运转。于是，各国政府和商业机构都越来越重视互联网基础设施的建设，不断加大对互联网基础设施的投入。互联网的接入方式也越来越多，带宽越来越大，速度越来越快。同时，互联网在可靠性和稳定性方面也越来越高，抗灾难能力也越来越强。

事实上，SaaS 软件依赖互联网的特点已经不是一个问题。

1.3.2 数据安全性

传统软件中，用户的数据存放在自己的电脑上，或者存放在企业自己的服务器中，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

用户自己会维护数据的安全性。只要做好数据的备份工作，一旦出现重大问题，也可以很容易将数据恢复到上次备份时的状态。对于用户来说，数据由自己掌握可能会感觉安全许多。

而 SaaS 软件的数据都是存放在软件商的网络服务器中的，用户对这些数据直接控制的能力有限。用户对将数据交给 SaaS 软件商是很担心的，一旦数据出了问题，用户将什么也干不了。用户的这种担心是完全可以理解的，特别是重要的商业数据，一旦丢失将对企业带来不可估量的损失。

要解决 SaaS 数据的安全性问题，要求 SaaS 软件商必须提供高可靠性的数据存储环境。妥善保管用户的数据，保证用户数据的可靠和安全，是 SaaS 软件商的天职。SaaS 软件商不但要对用户数据的安全做出承诺，关键还要能有实实在在的措施来保障数据的安全。只有这样，用户才敢将数据放到 SaaS 软件商的服务器上。

一般来说，有实力的 SaaS 软件商对用户数据安全的问题是非常重视的。SaaS 软件服务器所在的机房都非常专业，环境稳定，设施齐全。SaaS 软件的数据库一般都配备有双机热备份系统，实时备份用户数据。除非两台数据库服务器同时损坏，否则，用户的数据就总是安全可靠的。

此外，SaaS 软件商还有严格的定期备份制度。每隔一定时间就会将用户的数据从运行态的数据备份到磁带及光盘等，并进行归档或永久保存。一些有实力的 SaaS 软件商还能提供异地灾备服务，就是同时在相距很远的两个地理位置保留用户的数据备份。这样，即使发生重大的自然灾害，如火灾和地震等，都能保证用户数据的安全。

大多数情况下，只要用户参观过 SaaS 软件商的机房及数据库备份设施等，一般都愿意将自己的数据交给 SaaS 软件商来保存。因为，SaaS 软件商可以提供比用户自身更加安全可靠的数据存储服务。SaaS 软件商也只有大力投入数据存储方面的软硬件设施，规范和提高数据存储服务的质量和安全指标，才是解决用户担心数据安全问题的关键。

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

1.3.3 数据保密性

软件的数据不但需要安全的存储环境，还需要严格的保密机制。传统软件的数据都是由用户自己保存，所有的商业机密也都由企业自己掌控。只要企业做好自身的商业机密管理，就不存在泄密问题。

SaaS 软件就不同了。SaaS 软件的数据保存在 SaaS 软件商的数据库服务器中，SaaS 软件商完全可以知道用户的所有信息。也就是说，用户一旦开始使用 SaaS 软件，用户在这个软件中输入的所有商业机密也都完全暴露给了 SaaS 软件商。

特别是对于一些非常核心的商业机密，如财务机密、客户资料和真实价格等，一旦泄露给竞争对手，将对企业产生致命的伤害。这也是用户在使用 SaaS 软件时，不得不慎重考虑的主要原因，也是 SaaS 软件模式推广的最大障碍。

说实在的，要解决这个问题并不太容易。目前，也只有从两个方面去解决问题。一个是影响和改变用户的观念，另一个是加强 SaaS 软件商自身的信用建设。

影响和改变用户的观念是一个漫长的过程，要让用户都相信 SaaS 软件模式是未来软件的发展趋势也并非一件容易的事情。人们并非一开始就愿意将自己的钱存到银行里，同样，人们也并非一开始就愿意将数据存到 SaaS 软件商那里。

其实，用户的许多隐私数据都存到了网上。比如，电子邮件也包含大量的隐私数据，但似乎没有人去担心邮件服务器厂商会偷窥自己的电子邮件信息。随着互联网应用的不断深入，会有越来越多的用户隐私和商业秘密会保存到网络中，这是无法阻挡的潮流。这也是需要整个 SaaS 软件业共同来推进 SaaS 软件的普及才行。

SaaS 软件商在加大市场宣传的同时，更要注意使用软件的方便性。也许用户刚开始并不打算用 SaaS 软件来管理涉及商业机密的核心业务，不过只要 SaaS 软件做得足够方便，用户就会渐渐接受使用 SaaS 软件的事实。当用户发现其担心的泄密问题并没有发生的时候，也就会对 SaaS 软件商越来越信任。

我们认为加强 SaaS 软件商自身信用建设的方式是最有效的。SaaS 软件商必须制定

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

并实施一套严格的用户商业机密保护措施，加强对员工的职业道德管理，真正将用户商业机密的保护放到第一重要的位置。这就要在广大用户中树立崇高的信用，而 SaaS 软件商有了信用的品牌，用户才会放心地将自己的数据交给你去保管。

SaaS 软件商要树立自己的信用也是需要用实力来保障的。一般来说，较大的 SaaS 软件商比小的 SaaS 软件商更有实力，也就意味着信用也更有保障。这就有点像储户对银行的看法一样，越大的银行实力越雄厚，也就更有信用保障。用户选择 SaaS 软件商就像选择银行一样。

1.4 SaaS 成功案例

1.4.1 Salesforce.com

Oracle 的一位副总裁偶然使用了 Amazon 公司网站的在线订购，他意识到在线软件的应用可能会取代传统软件，成为软件行业的下一个潮流。经过多年的准备，他于 1999 年创办了 Salesforce 公司，专门从事在线软件的开发和销售。

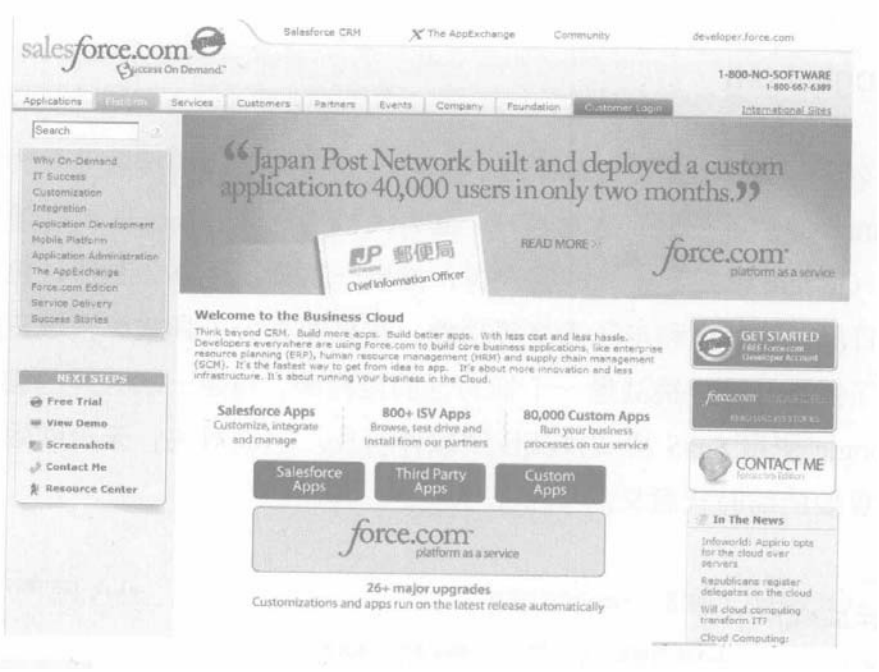
Salesforce 公司瞄准客户关系管理（CRM）应用，开发了在线的 CRM 应用软件。随后，在长期的在线 CRM 开发和销售中，Salesforce 逐步确立了在线软件市场的霸主地位。事实上，Salesforce 已经从一家在线 CRM 软件提供商，转化为一个全面的 SaaS 软件租用服务提供商。

长期的产品升级已经使 Salesforce 成为一个庞大的标准和体系，它本身产品功能的每一个升级都为它赢得更多的使用者。另外，任何符合它的接口标准的在线应用服务开发商都有可能成为它的合作伙伴。对于 Salesforce 这种在线软件服务商来说，没有一次程序的优化被浪费。在线软件服务商都将不断优化软件本身，这使得 SaaS 软件不会像传统软件那样，在达到一定功能后裹足不前。

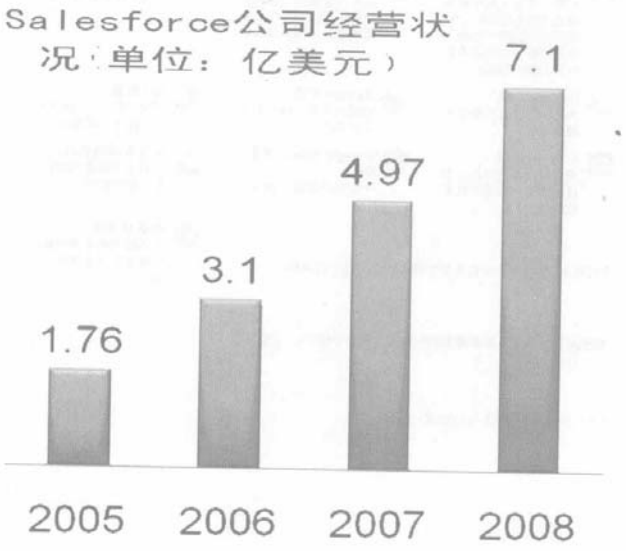
Salesforce 每发布一个新的功能，它的分布在全球几万个企业的近百万使用者都可能成为新功能的受益者。这些使用者发觉软件租用服务商在不断优化产品，不断地感受

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

到在线软件的新鲜感。这种不断升级的产品更适合以这种在线租用的方式提供给客户，客户以非常廉价的按月按用户的付费方式享受的是永远全新的软件。没有用户会质疑这种方式是局限的，难以接受的。



Salesforce 在商业上取得了巨大的成功。到目前为止，Salesforce 连续三年保持 50% 的增长速度。2008 年预期收入在 7.1 亿美元。总市值高达 60 亿美金。而最新的财务数据表明，2008 年的收入已经超过 10 亿美元，Salesforce 已经进入高速成长期。



Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目前，Salesforce 将其战略重点放到了平台开放和行业标准的制定上。其战略目的很清晰，就是要用开放和标准两大关键因素来引导和左右 SaaS 行业的发展，巩固其在 SaaS 行业的领先地位。

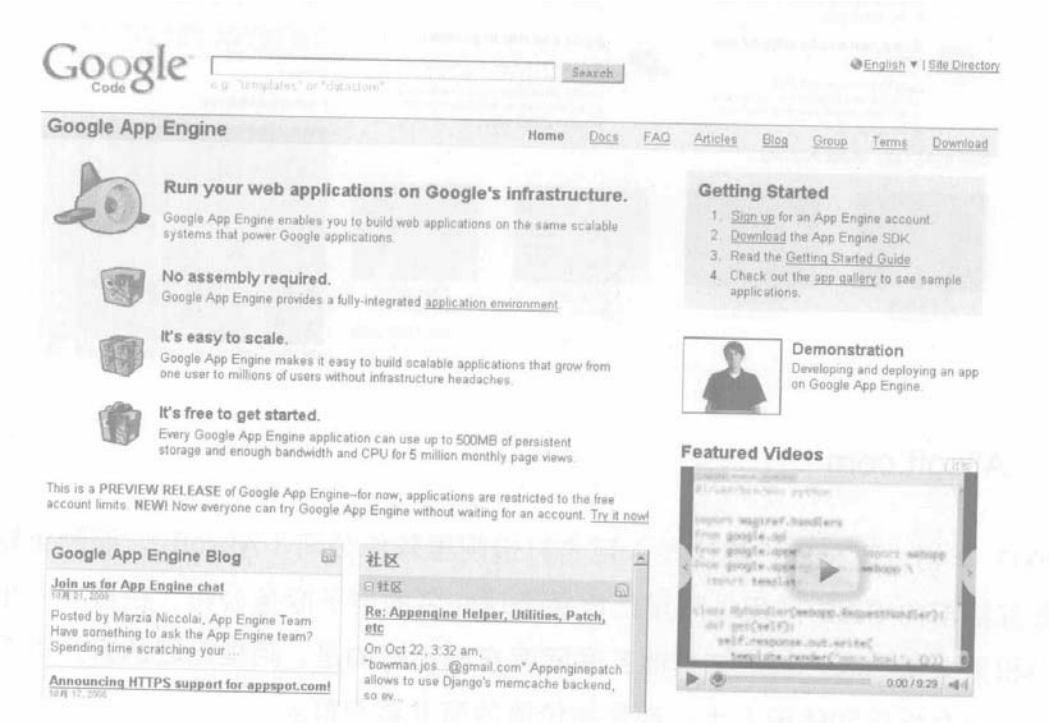
1.4.2 Google.com

Google 公司是 SaaS 行业的先锋之一。Google 以其核心的搜索业务为根基，依靠个性搜索和 Gmail 等产品积累了大量客户。据有关机构估计，Google 的注册用户数已经接近 1000 万。Google 依托其庞大的客户群，推出一系列在线办公套件，包括电子邮件、即时通讯、日历、字处理和电子表格等功能，在桌面办公工具软件领域正式向微软垄断的市场发起了挑战。这个挑战是一个标志性的里程碑，也是一次大胆的尝试。其重大意义在于，Google 使用 SaaS 模式挑战传统软件行业。这次对决，无论胜败都对软件行业的发展具有重要的跨时代意义。



互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

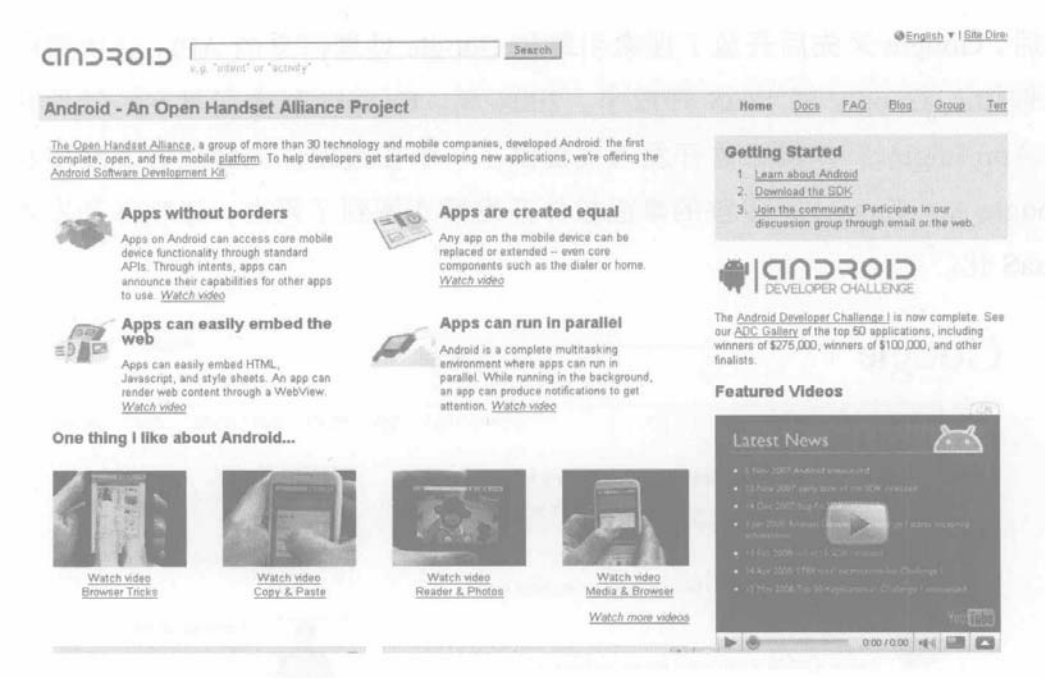
随后，Google 又先后开放了搜索引擎和 Google 地理信息的 API，让大量的第三方开发机构加入 Google 的 SaaS 开发中。2008 年，Google 发布了其在线软件开发平台 Google App Engine，向第三方开发人员提供了一个在线软件的集成开发环境和运行环境。Google App Engine 将传统的桌面软件开发模式搬到了网上，让软件开发本身也实现了 SaaS 化。



2008 年，Google 发布的智能手机 Android，标志着 Google 将其产业发展战略延伸到了通讯行业。Android 手机不但提供了丰富的移动商务应用，更重要的是它同时也是一个强大的手机 SaaS 软件开发和运行的平台。因为，未来的手机绝非一个移动电话，而是一台功能强大的手持计算机。Google 大力发展手机战略，无疑让 Google 在 SaaS 行业的竞争中占据了领导地位。

Google 的各种 SaaS 开发平台只是 Google 云计算战略的具体表现，Google 将未来目标锁定在深层次的互联网应用开发和推广上。Google 凭借其庞大的互联网资源、广泛的用户基础，以及雄厚的资本支持，必将成为全球 SaaS 行业发展中极其重要的力量。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



1.4.3 Alisoft.com

2007 年初，阿里巴巴集团投入巨资打造阿里软件公司（Alisoft），而阿里软件的核心业务直接指向 SaaS 应用。阿里巴巴是全球知名的电子商务公司，在多年的电子商务营销中积累大量的客户群。与其他互联网客户群不同的是，阿里巴巴的客户群没有小孩和学生，没有娱乐和休闲人士，都是高价值的商业客户群。

阿里巴巴集团深刻地意识到这些商业客户群的巨大价值，同时又敏锐地觉察到整个软件产业即将接受一场 SaaS 革命的洗礼。于是，阿里巴巴集团决定向第三方软件商开放整个阿里巴巴电子商务的所有资源，投资打造阿里软件互联平台，让众多的第三方软件上都加入到 SaaS 软件开发中，共享由此带来的利益。

阿里软件在成立后的短短一年多时间，打造了阿里软件互联平台——AEP 平台。该平台首次实现了接入、运行、推广、销售、计量、支付和维护等一体化运营平台。这既方便了许多想进入 SaaS 软件开发的软件厂商和个人，又同时为阿里巴巴的客户群提供了可以选择的丰富应用。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



为了给众多第三方软件商做示范，阿里软件在自身的平台上开发了中小企业外贸管理软件、在线财务软件和网络营销软件等几个典型示范应用。这几个应用在阿里巴巴的客户群中销售情况非常好，以至于阿里软件当年的 SaaS 软件销售额一举超过国内其他 SaaS 软件销售额的总和。阿里软件迅速成为中国 SaaS 软件产业的龙头企业，并带动了一大批第三方软件加盟阿里软件互联平台。



Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目前，阿里软件相继开放了 B2B、淘宝、支付宝和阿里妈妈等电子商务资源的 API 接口。这些接口以 Open API 方式提供，支持 REST 和 WebService 等多种调用形式。第三方软件商可以利用这些 API 来鉴别用户身份，对 SaaS 应用进行计量，并实现 SaaS 应用的收费等。

未来的阿里软件互联平台将更加开放，不但会开放 API，还将开放数据（Open Data）和开放集成界面（Open UI）。其发展方向是支持基于 SaaS 平台的多方应用集成和大规模定制，以满足广大用户的个性化需求。

1.4.4 其他 SaaS 案例

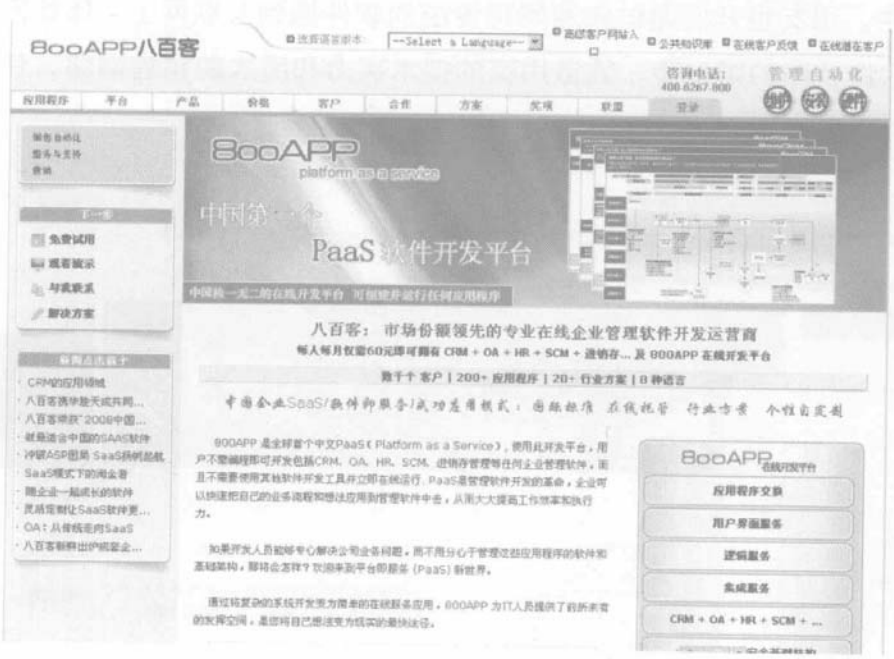
Xtools 是首批试水 SaaS 软件模式的国内公司。和 Salesforce 一样，Xtools 重点推广的是在线客户关系管理系统（CRM）。Xtools 抓住了中国用户的特点，一点点积累适合国内中小型企业的客户关系管理业务经验，打造了一款很实用的 CRM 系统。



800app.com 的中文名叫八百客，也是国内知名的 SaaS 软件厂商。与 Xtools 一样，重点是开发和运营在线客户关系管理系统（CRM）。除主要的 CRM 产品之外，八百客还将产

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

品线延伸至办公软件和进销存软件。这些丰富的产品线，也为八百客赢得了不少的客户。



SaaSBB 是神州数码和 Softbank BB 公司共同出资设立的 SaaS 专业软件公司。其产品线涵盖客户关系管理、企业业务管理和办公管理软件等多方面。除自身产品开发外，SaaSBB 也开始打造 SaaS 运营平台。



Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

伟库网（wecoo.com）是用友软件公司的 SaaS 软件门户。在传统软件向 SaaS 软件转变的大潮中，用友也开始通过伟库网将传统的软件搬到互联网上。伟库网仅仅是用友发展 SaaS 软件战略的第一步，凭借用友的技术实力和庞大的销售网络，伟库网的发展前景也是相当可观的。



友商网（youshang.com）是金蝶软件公司推出的在线软件门户网。金蝶公司作为国内传统软件的龙头企业之一，自然不甘在 SaaS 软件模式上落后于人。尽管目前的友商网还主要以提供在线财务软件服务为主，但不排除金蝶公司在未来会将其企业管理软件进行 SaaS 化。

由此可见，软件的 SaaS 化是软件产业不可阻挡的潮流，未来的软件市场将在 SaaS 软件营销方面展开激烈的争夺。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



1.5 小结

从计算机诞生的那一天起，就产生了软件的概念。软件一开始是计算机的附属品，经过产品化的发展历程之后，走向 ASP 模式，而最终成为了一种服务，由此诞生了 SaaS 的概念。SaaS 将软件从产品变成了服务，从而改变软件从开发、推广、销售直到维护的整个商业模式。

SaaS 与 ASP 有着剪不断理还乱的关系。SaaS 起源于 ASP，但却从理论和实践两方面都远远超越了 ASP 的局限性。SaaS 实际上更关注软件的服务化，以及由此带来的软件模式的变革，并不在乎软件服务是以何种技术方式提供的。随着 SaaS 软件的迅猛发展，区分 SaaS 与 ASP 已经变得毫无意义。

SaaS 软件可以给用户带来许多优点。SaaS 软件无须安装拿来即可使用，用户也无须维护。SaaS 软件先天防毒，也给用户省去不少麻烦。用户使用 SaaS 软件的风险较低，成本会减少很多。只要有网络的地方就可以使用 SaaS 软件，而且用多用少完全由用户自己决定。

SaaS 架构设计
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

SaaS 也给软件商带来了新的机遇。SaaS 可以减少软件商的销售成本，还能降低维护软件和客户服务的成本。更重要的是，SaaS 带给软件厂商一种稳健的商业经营模式。这种稳健的商业模式可以让软件商与客户之间形成良性的循环，能够真正达到软件开发商和软件使用者的双赢目标。

SaaS 软件模式也有其自身的缺点。主要缺点包括依赖互联网，出现数据安全问题影响很大，在数据保密性方面也难有有效的技术手段来保证。加强互联网基础设施的建设，构建高可靠的数据库环境，以及提高 SaaS 软件厂商的信用，都是解决这些缺点的办法。

SaaS 软件模式已经有许多成功的案例。国外的如 Salesforce 和 Google 等，在 SaaS 领域都取得了巨大的成功。国内的 SaaS 产业虽然起步较晚，但国内 SaaS 软件的发展也已初露锋芒，特别是阿里软件（Alisoft）更是发展成为国内最大的 SaaS 平台商。其他如 Xtools、800app、SaaSBB、伟库网和友商网等，纷纷加入 SaaS 市场的争夺。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第2篇

SaaS 应用架构

SaaS 不仅仅是商业模式上的创新，其要取得成功更依赖于技术架构的创新。

传统软件和 SaaS 应用架构上的差别是什么？

开发 SaaS 应用将面临的挑战有哪些？

如何实现高性能、可配置、可伸缩的 SaaS 应用？

作为新进入 SaaS 领域的开发商，又该如何逐步成熟？

刚刚接触 SaaS 的读者，可能会对 SaaS 应用架构方面产生一系列的问题。接下来的第 2 部分将通过 SaaS 成熟度模型描述 SaaS 应用逐步成熟，逐步实现高性能、可配置性、伸缩性的过程，并对 SaaS 应用安全、离线应用做专题介绍。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 2 章

SaaS 成熟度模型

2.1 SaaS 的规模效应 …… 33
2.2 SaaS 成熟度模型分级 …… 36
2.2.1 Level 1：定制开发 …… 36
2.2.2 Level 2：可配置 …… 37
2.2.3 Level 3：高性能的多租户架构 …… 39
2.2.4 Level 4：可伸缩性的多租户架构 …… 41
2.3 如何选择合适的 SaaS 成熟度模型 …… 43
2.3.1 产品所面向的客户群的特征与需求 …… 43
2.3.2 产品的租户数量级别 …… 44
2.3.3 团队的开发能力与愿意付出的开发/改造成本 …… 45
2.4 SaaS 软件的成熟度模型的渐进步骤 …… 46
2.5 小结 …… 47

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2.1 SaaS 的规模效应

SaaS 软件相对于传统软件，将原本由软件使用者所承担的软硬件、网络、系统维护的费用，转成支付给 SaaS 服务提供商的租用费用。而对于 SaaS 服务提供商而言，依然要承担相应的软硬件、网络、系统维护等费用。除了降低软件使用者的一次性投入成本，将一次性的投入转变成按时间、需求的逐步投入，单纯的责任转换并不会创造出额外的价值。因此，SaaS 软件要降低企业综合使用成本，最主要的手段就是发挥 SaaS 的规模效应。

传统模式与 SaaS 模式的部署模型对比如图 2-1 所示。

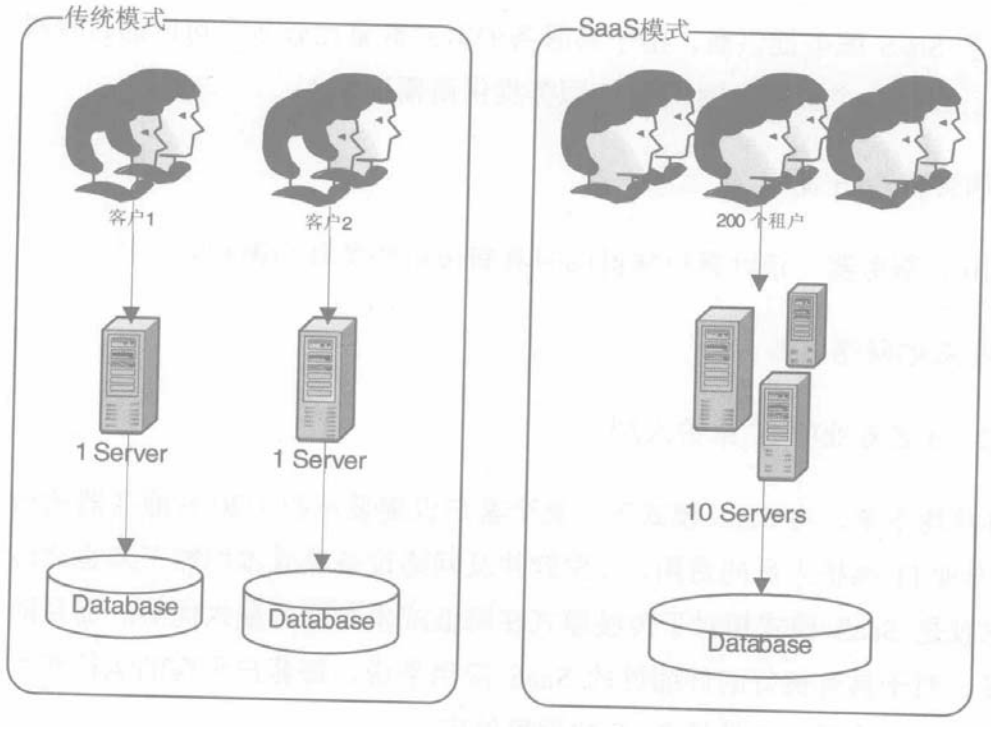


图 2-1　传统模式与 SaaS 模式的部署模型对比

例如，对于传统的软件模式，一个客户需要使用一款 CRM（Customer Relation Manager，客户关系管理）软件，需要投入：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 购买软件/定制开发软件；
- 至少 1 台服务器；
- 相应的网络设备；
- 至少 1 名专业的 IT 维护人员；
- 如果对于负载均衡和可用性有较高的要求，可能还需要额外的服务器投入。

在 SaaS 模式下，客户则仅需从 SaaS 服务提供商那里购买相应的 CRM 软件服务即可，软件开发、硬件（服务器及网络设备）、软硬件的维护，以及实现高可用性的设备都通通由 SaaS 服务提供商来搞定。

而对于 SaaS 服务提供商，由于其服务的客户数量比较多，可以通过规模效应来降低其成本。以 200 个客户为例，SaaS 服务提供商需要投入：

- 购买软件/开发软件；
- 10 台服务器（该计算机集群同时具有良好的负载均衡和可用性）；
- 相应的网络设备；
- 2～3 名专业的 IT 维护人员。

这样平均下来，在 SaaS 模式下，每个客户仅需要承担 1/20 台服务器的费用，以及 1/100 名专业 IT 维护人员的费用，开发软件及网络设备等成本均摊下来也比传统模式小很多。这就是 SaaS 模式相对于传统模式在降低成本方面的最大优势。而且随着客户数量的增多，对于具有良好的伸缩性的 SaaS 应用来说，每客户平均的运行和维护成本将可以得到进一步降低，这就是 SaaS 的规模效应。

传统软件与 SaaS 软件 200 个客户的综合成本对比如表 2-1 所示。传统模式和 SaaS 模式下的规模效应对比如图 2-2 所示。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 2-1　传统软件与 SaaS 软件 200 个客户的综合成本

比 较 项	传统软件 200 个客户的综合成本	SaaS 软件 200 个客户的综合成本
软件开发成本	1 ~ 200 个软件的开发成本，具体取决于客户需求的异同，以及软件可配置性强弱	1 个软件的开发成本
服务器成本	200 台服务器	10 台服务器
网络设备成本	200 套	5 ~ 10 套
IT 维护人员成本	200 人	2 ~ 3 人

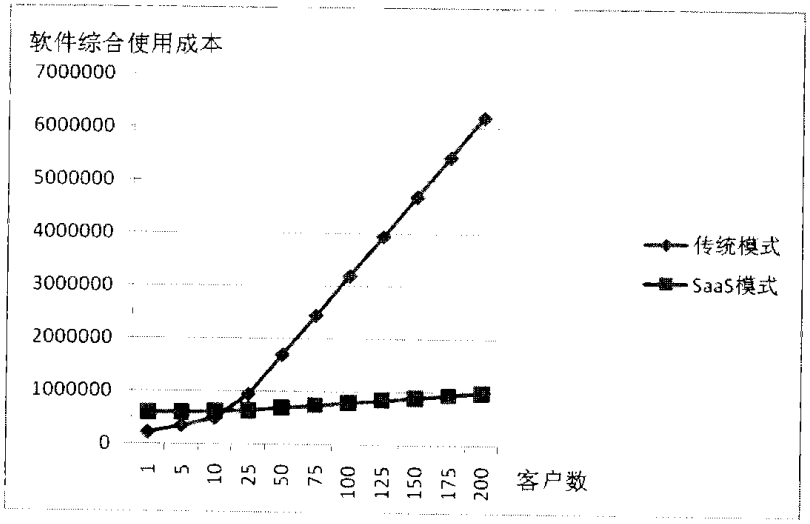


图 2-2　传统模式和 SaaS 模式下的规模效应对比

而如何最大化 SaaS 的规模效应，就不仅仅是商业模式的问题，更是一个应用架构的问题。SaaS 成熟度模型的高低，就是 SaaS 规模效应高低的最重要体现。

对于 SaaS 应用的架构师而言，尤其是从传统软件转型到 SaaS 的架构师，遇到的首要挑战就是多租户的思维转变。传统软件的销售模式决定了软件的每一个运行实例都服务于一个客户，因此对于性能、可配置性、伸缩性方面的处理策略都是针对单一客户所设计的。而对于 SaaS 应用，为了最大化 SaaS 的规模效应，一般采用的都是多个租户共享一个运行实例的架构（Multi-Tenant 架构，即多租户架构）。如何在 Multi-Tenant 架构下，实现 SaaS 应用的高性能、可配置性、伸缩性的目标，就是 SaaS 应用架构师所面临的最大挑战。这些也构成了判定 SaaS 成熟度模型的主要特性。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2.2 SaaS 成熟度模型分级

根据 SaaS 应用是否具有可配置性、高性能、可伸缩性的特性，SaaS 成熟度模型被分成四级。每一级都比前一级增加以上三种特性中的一种（见表 2-2）。

表 2-2 四级成熟度模型所具备的特性比较

	可配置	高性能	可伸缩
Level 1	×	×	×
Level 2	√	×	×
Level 3	√	√	×
Level 4	√	√	√

2.2.1 Level 1：定制开发

在 Fred Chong 等的《Architecture Strategies for Catching the Long Tail》一文中，将这种最初级的成熟度模型定义为 Ad Hoc/Custom，即特定的/定制的，不过就应用架构而言，对于传统的软件开发者，我们认为“项目型软件开发”正好符合定制开发的特征，这也许能更容易让大家理解。因为，对于最初级的成熟度模型，技术架构上真的跟传统的项目型软件开发或者软件外包没有什么两样。有一个客户项目，就按照客户的需求来定制一个版本，每个客户的软件都有一份独立的代码。不同的客户软件之间可以共享和重用的也就只是少量的可重用的组件、库以及开发人员的经验。

最初级的 SaaS 应用成熟度模型在应用架构上，跟传统的项目型软件应用架构几乎没有差别。应该说最初级的 SaaS 应用成熟度模型与传统模式的最大差别在于商业模式，即软硬件以及相应的维护职责由 SaaS 服务提供商负责，而软件使用者只需按照时间、用户数、空间等逐步支付软件租赁使用费用即可。

在应用架构上，最初级的 SaaS 应用成熟度模型与 20 世纪末的第一次互联网浪潮中曾经一度流行的 ASP（Application Service Provider）基本是一致的。在这种模型下，软件服务

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

提供商为每个客户定制一套软件，并为其部署。每个客户使用一个独立的数据库实例和应用服务器实例。数据库中的数据结构和应用的代码可能都根据客户需求做过定制化修改。

第一级成熟度模型如图 2-3 所示。

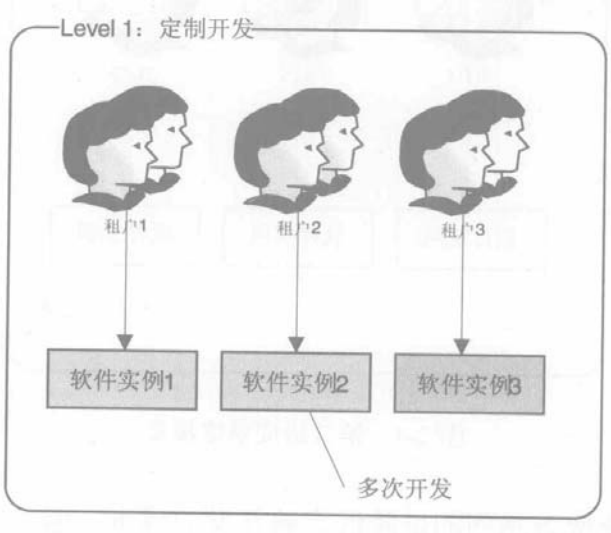


图 2-3　第一级成熟度模型

实际上，传统的 C/S、B/S 软件在商业模式上做出相应的改变，由软件开发商来部署软件，并按需付费，而应用架构上几乎不用变化，这就完全符合最初级的 SaaS 成熟度模型。虽然这在 SaaS 应用的成熟度模型中还是最初级的，还不具备高成熟度 SaaS 模型所具有的许多特性。但是，相对于传统软件模式，SaaS 应用提供商通过整合软硬件资源，在降低软硬件以及专业化的维护成本方面取得了一定的规模效应，从而在一定程度上降低用户使用软件的综合成本。

2.2.2　Level 2：可配置

不过，最初级的成熟度模型显然不是良好的 SaaS 成熟度模型。由于对于每个客户都需要做特定的代码级定制，都需要独立部署，其每个客户的定制开发成本、软硬件以及运行维护成本基本上随客户数量增加而按比例增加。当这种模式下的应用达到一定的规模之后，想要再进一步扩大应用的规模，基本上就只能依赖于人肉战术。这大概也是国内很多软件企业一直都是惨淡经营的主要原因。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第二级成熟度模型如图 2-4 所示。

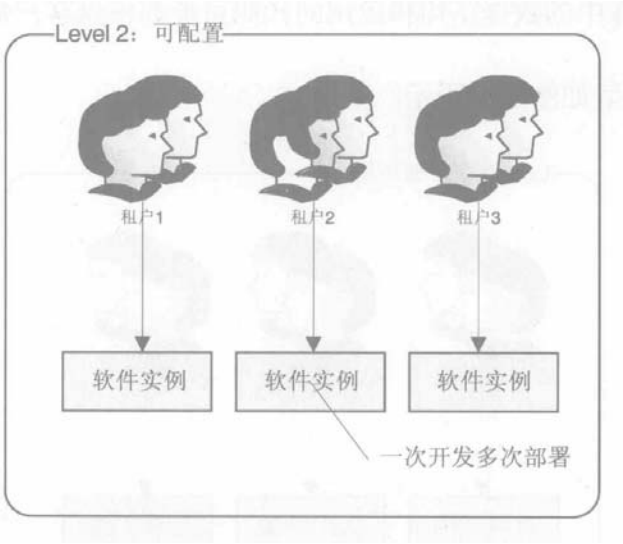


图 2-4　第二级成熟度模型

这时候，我们首先应该想到的是降低定制开发的成本。因为，在这种情况下，定制成本成为了每个客户付出成本中的最大部分。降低软件的定制开发成本大概也是传统软件开发发展的一个趋势。很多项目型的软件公司，在达到一定的积累之后，也会考虑转型开发通用型的产品。当然，为了满足不同客户的不同需求，通用型的产品通常具有可配置性，客户可以通过简单的配置，让通用型的软件产品能够满足自己的一些个性化需求，而通用型产品的可配置性的强弱，通常也是衡量软件是否通用的一项重要指标。

SaaS 成熟度模型中同样如此。第二级成熟度模型相对于最初级的成熟度模型，增加了可配置性。希望通过不同的配置来满足不同客户的需求，而不需要为每个客户进行特定定制，以降低定制开发的成本。

但是，在第二级成熟度模型中，软件的部署架构没有发生太大的变化，依然是为每个客户独立部署一个运行实例。只是每个运行实例运行的是同一份代码，通过配置的不同来满足不同客户的个性化需求。

满足第二级成熟度模型的软件很多。目前，互联网上很多 B2C 网上商城系统、论坛系统都提供此类服务。一旦客户购买相应的软件，该软件提供商就会负责其软件的硬

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

件部署、网络环境以及后续的维护。通过软件本身提供的配置功能可以满足不同客户特定的需求，而客户则仅需按年支付相应的服务费即可。

可配置性的比较通用的实现方式，就是通过 MetaData（元数据）来实现。但是对于传统的并没有基于元数据来设计的应用架构，要改造成符合第二级成熟度模型的应用架构，需要从底层架构就开始做比较大的改造，系统复杂度较高。

2.2.3 Level 3：高性能的多租户架构

其实，在应用架构上，第一级和第二级的成熟度模型与传统软件没有多大差别，只是在商业模式上符合 SaaS 的定义。由于其应用架构是设计成给每个租户部署一个运行实例的形式，必然导致硬件及运行维护成本很难产生较大的规模效应。毕竟一台服务器上能运行的应用实例肯定比较有限，一个人能够维护的应用实例也会比较有限。因此，多租户单实例的应用架构才是通常真正意义上的 SaaS 应用架构，也就是我们通常所说的 Multi-Tenant 架构。多租户单实例的应用架构可以有效地降低 SaaS 应用的硬件及运行维护成本，最大化地发挥 SaaS 应用的规模效应。

第三级成熟度模型如图 2-5 所示。

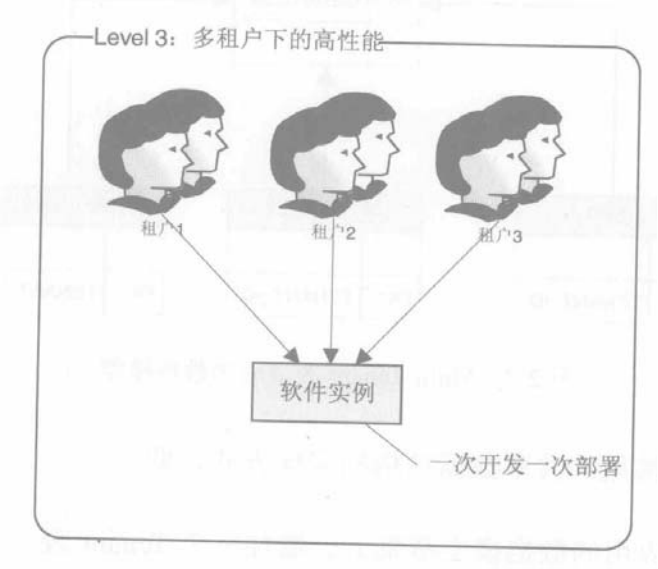


图 2-5 第三级成熟度模型

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

目前互联网上的大部分 SaaS 应用，尤其是用户量比较大的互联网应用，其应用架构大多符合第三级成熟度模型。例如，Email、blog 等服务，虽然归入 SaaS 类应用有些牵强，但其在应用架构上却是符合 Multi-Tenant 架构的。

要实现 Multi-Tenant 架构的关键是通过一定的策略来保证不同租户间的数据隔离，确保不同租户既能共享同一个应用的运行实例，又能为用户提供独立的应用体验和数据空间。在客户需求差别不大，客户数量也不是特别大的情况下，将一个第一级/第二级成熟度的应用改造成符合 Multi-Tenant 架构的应用并不会太复杂。在改造方案上，通常会有独立数据库、共享数据库独立数据结构、共享数据结构等多种实现方案。在接下来的第 3 章中，我们将一一详述这几种实现 Multi-Tenant 架构的方案。

图 2-6 和图 2-7 展示了一个 CRM 系统在实现 Multi-Tenant 改造前后的数据模型。



图 2-6　初始数据模型

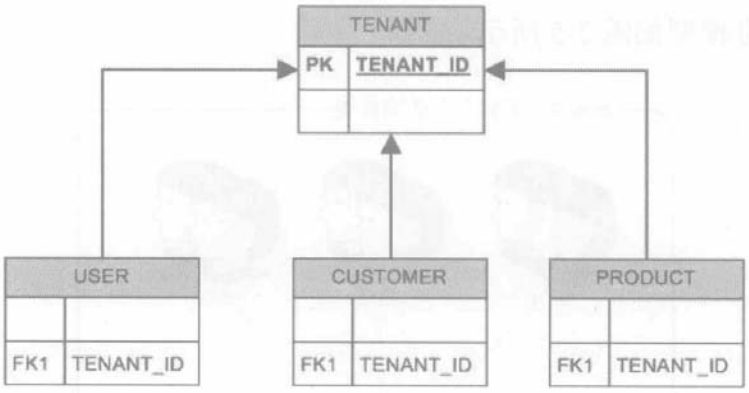


图 2-7　Multi-Tenant 改造后的数据模型

图 2-7 描述了最常见的共享数据结构的实现方式，即：

- 在传统企业应用的数据模型基础上，增加一个 Tenant 表（或者类似作用的表，例如公司表），用于描述租户信息。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 在大部分与租户有关的业务数据表中增加 TENANT_ID 字段。
- 数据模型改造完成后，还需要改造登录逻辑，在会话中记录用户所属的 TENANT_ID。
- 在业务数据查询过滤时，都增加上 TENANT_ID = ?过滤条件。

要实现基本的 Multi-Tenant 架构不难。不过与传统的企业应用不同，SaaS 应用更具备互联网应用的许多特征。与传统单实例的企业应用相比，其数据量和并发量都会有显著的增加。因此，对于基于传统应用改造的 Multi-Tenant 应用，还会面临大量的性能方面的挑战。第 4 章我们将会详细介绍 SaaS 应用在实现高性能方面的诸多最佳实践。这里很多的实践也可以说是互联网应用通用的性能优化的最佳实践。

除了面临性能方面的挑战，Multi-Tenant 应用在实现可配置性方面相对于传统企业软件也会有明显的差别。例如，数据模型的扩展方案相对于传统企业应用的实现方案就会有很大差别。第 5 章我们将会详述在 Multi-Tenant 架构下实现可配置性的方案。

2.2.4 Level 4：可伸缩性的多租户架构

在实现了多租户单实例的应用架构之后，随着租户数量的逐渐增加，集中式的数据库性能就将成为整个 SaaS 应用的性能瓶颈。如果我们不进一步考虑数据库的分区设计，则只能依赖于更强大的硬件设备来向上扩展（scale up）。当应用规模增长所带来的压力达到单一硬件设备的极限时，最终将导致 SaaS 应用架构难以满足低成本的运营需要。

第四级成熟度模型如图 2-8 所示。

因此，如何通过一定的策略来满足 SaaS 应用的水平扩展（scale out）的要求，成为 SaaS 应用架构设计需要解决的问题。即在用户数大量增加的情况下，无须更改应用架构，而仅需简单增加硬件设备的数量，就可以支撑应用规模的增长。不管用户多少，都能像单用户一样方便地实施应用修改。这就是第四级也是最高级别的 SaaS 成熟度模型所要致力解决的问题。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

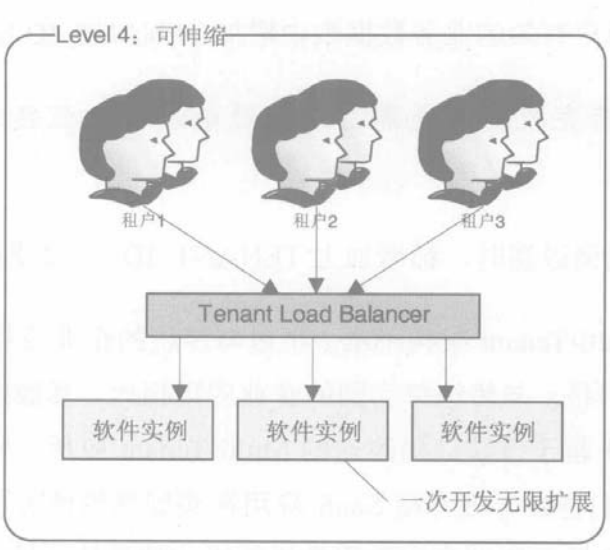


图 2-8　第四级成熟度模型

最高级的应用成熟度模型相对于第三级成熟度模型的变化，就是将 Multi-Tenant SingleInstance 系统扩展为 Multi-Tenant MultiInstance。并且，最终用户首先通过接入 Tenant Load Balance 层，再被分配到不同的 Instance 上。通过多个 Instance 来分担大量用户的访问，我们可以让应用实现近似无限的水平扩展。

要实现第四级成熟度模型，最复杂的就是针对原有单个 Instance 的数据库服务器，实现其数据的水平拆分。基于这种多实例的数据水平拆分方案的框架，SaaS 应用架构大致如图 2-9 所示。

在上述架构中，Tenant Load Balance 层会存放用户、租户与对应的 Instance 的映射关系。在用户登录后，即可通过该映射关系，将用户重定向到相应的 Instance。在 SaaS 应用初始阶段，租户数量比较少的时候，可能仅需部署 1 个 Instance，即可满足数千甚至上万的租户的需求。随着租户数量的进一步增大，该架构可以保证仅通过再多部署几个 Instance 来满足更多租户的使用。

在第 6 章中，我们将会针对如何实现 SaaS 应用的伸缩性方案这一问题，做详细的阐述。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

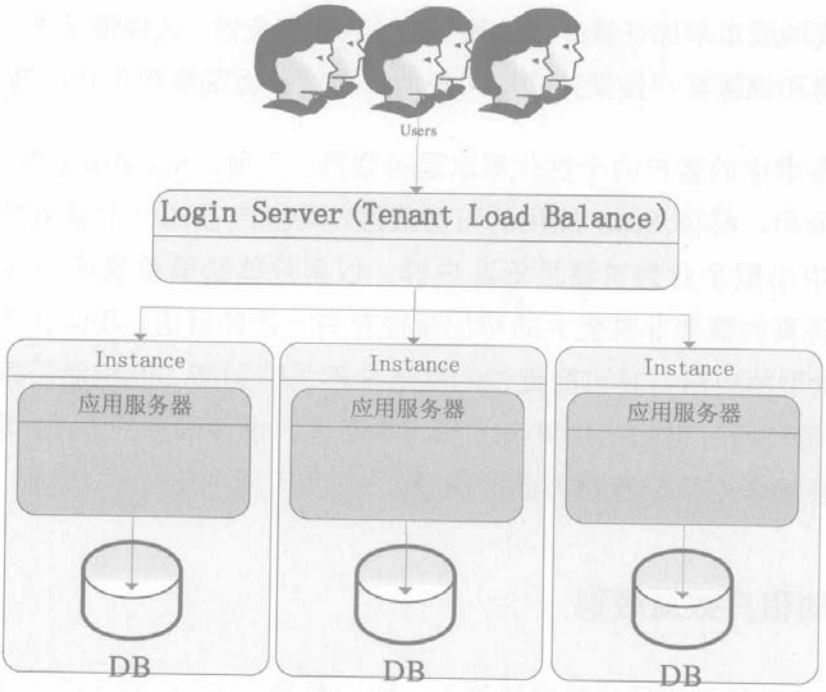


图 2-9　第四级成熟度模型 SaaS 应用架构

2.3　如何选择合适的 SaaS 成熟度模型

虽然从应用架构的角度，同时具备可配置性、高性能和可伸缩性的第四级 SaaS 成熟度模型是我们最为理想的应用架构。但是，综合商业需求、实现成本及复杂度等各方面考虑，并不是所有的 SaaS 应用都必须达到第四级成熟度模型。具体每个应用应该选择哪个级别的 SaaS 成熟度模型，需要综合考虑。需要考虑的因素可能包括：

- 你的产品所面向的客户群的特征与需求；
- 你的产品的租户数量级别；
- 你的团队的开发能力与你们愿意付出的开发/改造成本。

2.3.1　产品所面向的客户群的特征与需求

首先需要考虑你的客户是否愿意接受共享应用的模型。很多场景下，部分大客户可

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

能愿意花费更高的成本来保证其应用的独立性、数据安全性。这种情况下试图去解释不同租户的数据隔离和说服客户接受共享应用模型，可能会面临意想不到的商业上的困难。

然后还要考虑你的客户的个性化需求是否强烈。尽管，Salesforce 等一些在 SaaS 领域非常成功的公司，都将 SaaS 应用的可配置性作为其产品的一个突出的卖点。但是，我们依然对以中小型企业为首要目标客户群，以用户体验简单易用为突出竞争优势的 SaaS 应用，是否真的需要非常强大的可配置性存有一定的疑虑。我们认为至少有相当一部分的 SaaS 类型的应用，其可配置性的需求并不是特别高。而相对于基于元数据实现 SaaS 应用可配置性所付出的架构复杂度和成本而言，也许很多的 SaaS 类型的应用，并不需要过早过多地考虑可配置性方面的问题。

2.3.2 产品的租户数量级别

如图 2-10 所示是在不同成熟度模型下，租户数量与 SaaS 产品综合成本（包括开发和运行维护成本）之间的关系。从该图可以看出，成熟度更高的 SaaS 产品，在开发成本上会更高，但是在每租户的运行维护成本上会降低，因此租户数量是决定你选择怎样的成熟度级别的一个重要因素。

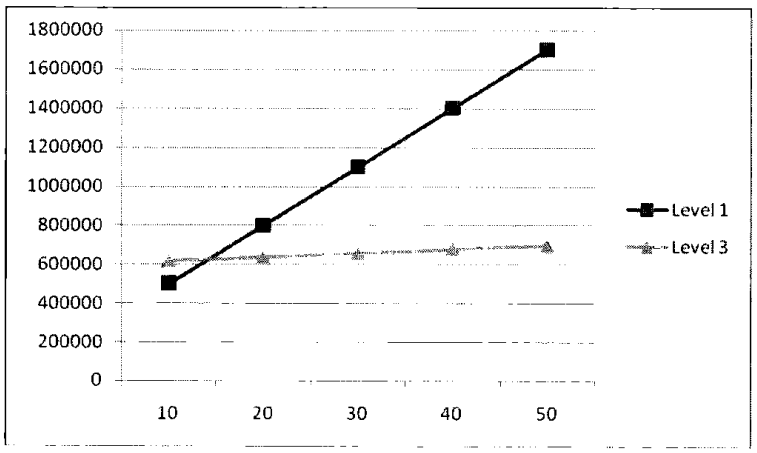


图 2-10　第一级与第三级成熟度模型在不同租户量下的综合成本比较

虽然，在大量的租户下，Multi-Tenant 架构的规模效应会得到显著体现。但是，当你的租户数量很小，而且你也认为你的目标用户群在短期内不会发生太大变化时，你就

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

需要在传统应用改造成 Multi-Tenant 架构所需要付出的成本与带来的收益之间进行权衡。我们的经验数据表明，租户数量小于 10 的时候，Multi-Tenant 架构所带来的收益并不大；而当租户量在大于 50 之后，Multi-Tenant 架构是 SaaS 应用的必然选择。

在伸缩性架构的实现上，图 2-11 描述了第三级成熟度模型和第四级成熟度模型在租户数量不断增长情况下综合成本的变化情况。当租户数量增加到一定程度以后（例如 10 000 以上），如果无法实现水平扩展，则仅能够通过向上扩展来提升系统容量和性能，这种情况下每租户的平均硬件成本会显著上升。租户数量进一步增加之后，甚至有可能不能支持。而在实现了伸缩性架构之后，虽然初始架构复杂度会增加，开发成本更高。但是，在超大规模的租户数量下，其水平扩展能力能够保证每租户的平均综合成本不会随租户数量的增加显著上升。我们的经验数据表明，租户数量在 10 000 以下的时候，可伸缩性不是 SaaS 应用首先需要考虑的问题；而当租户数量在 50 000 ~ 100 000 甚至更高的时候，可伸缩性架构是你的系统需要重点考虑的。

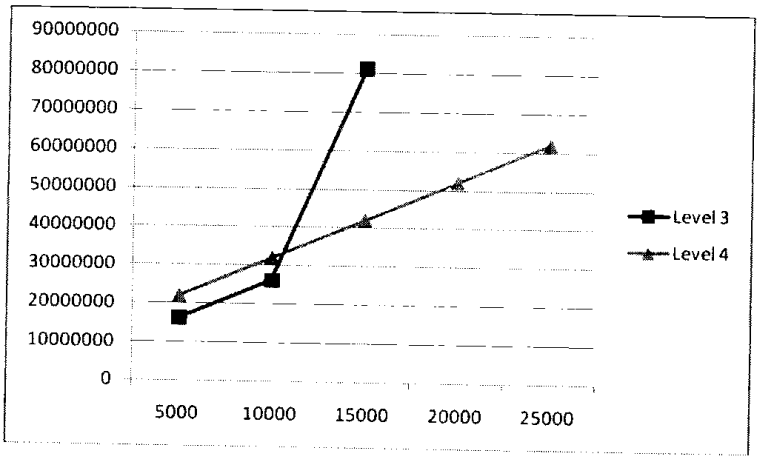


图 2-11　第三级与第四级成熟度模型在不同租户量下的综合成本比较

2.3.3　团队的开发能力与愿意付出的开发/改造成本

团队开发能力也是在选择合适的 SaaS 成熟度模型时所必须考虑的。对于 SaaS 架构经验不足的团队而言，我们更倾向于按照一定的顺序逐渐地实现 SaaS 成熟度模型所要求的三个特性。这一点就是我们在 2.4 节中所描述的内容。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

此外，更高的成熟度模型意味着更复杂的架构，也就意味着更高的开发/改造成本，这一点也是你的团队需要综合权衡的。

总而言之，并不是成熟度越高的 SaaS 应用就越好，越能够取得成功。这就跟并不是最好的技术就能取得成功一样，要根据具体的场景，选择最合适的技术，选择最适合的成熟度模型。没有最好，只有最适合。

2.4 SaaS 软件的成熟度模型的渐进步骤

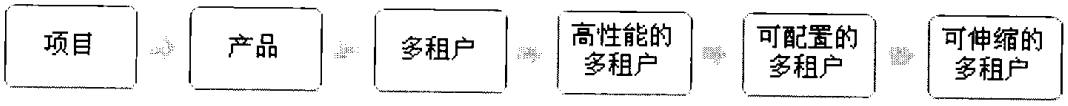
尽管，SaaS 成熟度模型的四个级别是按照配置性、多租户的高性能、伸缩性的顺序逐渐增加的。但是，这并不代表所有的 SaaS 应用都应该按照这个顺序来增加相应的成熟度特性。例如，就我们对于 SaaS 应用所面向的客户群体、商业需求、实现成本与复杂度的综合考虑，我们更倾向于按照如下的步骤来逐步增加成熟度。

- **实现多租户（Multi-Tenant）架构**：Multi-Tenant 架构是 SaaS 应用的基本特性，也是实现 SaaS 规模效应的基本要素。
- **实现多租户架构下的高性能**：高性能无论是对于客户体验，还是降低成本的角度都是至关重要的。在尚未实现伸缩性之前，优化和提升系统性能，更是提升你的系统客户规模/容量的最主要方法。
- **实现可配置**：客户的需求变化非常广，因此可配置性的范围也相当的广，能够满足所有客户需求变化的系统几乎不可能存在。一般情况下，要满足的客户需求变化越多，则系统的复杂度越大。因此，有效地抓住你的客户群体的需求变化，并针对这些变化进行相应的设计，实现可配置性，以取得可配置性与系统复杂度的平衡显得更加重要。
- **实现伸缩性**：伸缩性实际上也包含多个方面，包括应用层的伸缩性、数据库层的伸缩性等（具体内容在第 6 章会详细介绍）。一般来说，应用层的伸缩性更容易实现，是大部分 SaaS 应用所必备的特性（否则只有一台应用服务器，很难支撑起大量租户）。而数据库层的伸缩性，无论在实现的复杂度上，还是在需求方

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

面，对于很多的小型网站/SaaS 软件，都不是必需的。据我们所知，包括很多的大型互联网网站，也都还是采用集中式数据库。总之，伸缩性方面的考虑应该基于你目前系统容量与你未来一定时期内期望达到的系统容量。

正是基于以上考虑，我们在接下来的 4 章中，将通过两个大学生郭靖和杨康的创业故事来详细阐述逐步实现 SaaS 成熟度模型的方案和最佳实践。下面这个图是郭靖和杨康在创业历程中走过的每一个里程碑，也是 SaaS 软件成熟度模型渐进演化的过程。



2.5 小结

规模效应是衡量 SaaS 成熟度的重要特性，而多用户的高性能、可配置性、伸缩性则是成熟的 SaaS 应用需要具备的 3 种特性。根据 SaaS 应用是否具备这 3 种特性，我们描述了 4 种成熟度模型：

（1）定制开发——项目式的软件开发。

（2）可配置——产品式的软件开发。

（3）高性能的多租户架构。

（4）可伸缩性的多租户架构。

虽然成熟度模型是递进的，每一个级别的成熟度模型都相对于上一个级别，增加了以上 3 种特性中的一种。但是，我们也并不是建议所有的应用开发者一味地追求最高级别的成熟度模型。每个人需要根据自己的商业、技术架构等多个角度来综合考虑，做出最终的选择。

本章只是对于 SaaS 成熟度模型的概述性的指引，接下来的第 3～6 章我们将一一详述 SaaS 软件架构是如何逐步成熟起来的。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第3章 构建 Multi-Tenant 应用

3.1 第一阶段：做项目 …… 49
3.1.1 发现商机 …… 49
3.1.2 “4+1”视图 …… 50
3.1.3 设备托管 …… 58
3.2 第二阶段：做产品 …… 59
3.2.1 设备共享 …… 59
3.2.2 创业之旅 …… 61
3.2.3 可配置化 …… 62
3.3 第三阶段：多租户 …… 63
3.3.1 成长的烦恼 …… 63
3.3.2 如何转化成 SaaS 多租户模式 …… 65
3.3.3 SaaS 多租户设计 …… 70
3.3.4 SaaS 多租户模式的威力 …… 72
3.4 小结 …… 73

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

郭靖和杨康是同窗挚友，现在读大学四年级，明年就要毕业了，想到即将走出校门，两人心中充满了期待和忐忑。

郭靖出生在北方城市，从小接触计算机，经常写个小程序什么的，无论搭建交友社区，还是帮别人建网站都不在话下，很早表现出在编程方面的过人天赋。进入大学后他更是延续了对 IT 技术的狂热追逐，在知名网站的 IT 频道有自己的技术博客，同时还是好几个论坛的版主。

杨康来自南方大都市，父母和亲戚不少都在做生意，规模并不大，但生意很红火。杨康从来都是个活跃分子，喜欢与人打交道，有他在的时候他永远是大家交谈的中心。他炒股、泡妞，在别人眼里好像有些不务正业，但他有独到的眼光，他也有他的梦想，那就是依托互联网成就自己的一番事业。

3.1 第一阶段：做项目



3.1.1 发现商机

这不，暑假到了，杨康又琢磨着干点啥。通过学弟欧阳克引见，杨康来到欧阳锋的公司，想通过自己的观察和感受多了解企业的现状和需求。欧阳锋的公司是做电子贸易的，客户来自全国各地，不一会工夫，欧阳锋和他的业务员就见了 5 个客户，中间还被好几个外地客户的电话打断。

杨康：欧阳叔叔，您精力充沛，以一当十啊！

欧阳锋：哪里啊，其实是没办法，现在客户多得都快管不过来了，光靠脑子是记不下来的，只能再请几个帮手了……

杨康发现了这其中的商机，为什么不用软件管理起来呢？

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

杨康：其实用不着，假设我们开发一套系统，管理所有的客户资料、商务行程和客户订单，那肯定可以大大提升效率。

欧阳锋：也许是个办法，你小子不是搞软件的吗，我就交给你好了，让克儿也一块帮忙。

杨康：没问题，我这两周就跟着您，先搞清楚公司的业务流程。

……

要做 CRM（Customer Relation Management）系统，一两个人是搞不定的，当然要拉上他的死党郭靖。三人分工，杨康负责商业需求，郭靖和欧阳克负责设计开发。

3.1.2 “4+1”视图

拿到需求后郭靖很是兴奋，以前的设计都是自己搞着玩的，这回终于可以面向真正的客户、大展拳脚了。事不宜迟，郭靖开始忙起来，想到要让欧阳克理解架构、尽快上手，他打算参考 RUP（Rational Unified Process）中的“4+1”视图模型（逻辑视图/过程视图/开发视图/物理视图+场景视图）来分析设计（见图 3-1）。

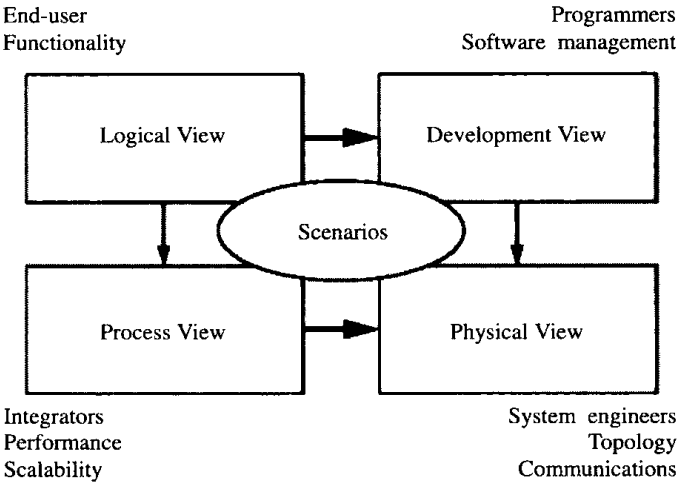


图 3-1 “4+1”视图

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

“4+1”视图最早由 Philippe Kruchten 提出，他在 1995 年的《IEEE Software》上发表了题为《The 4+1 View Model of Architecture》的论文，引起了业界的极大关注，并最终被 RUP 采纳，现在已成为架构设计的结构标准。五类视图需要考虑的问题不一样：

- **场景视图**

也叫用例视图，描述用户的业务场景，从用户的角度识别出业务需求，它是架构设计的起点和终点。

- **逻辑视图**

如果设计采用面向对象的方法，逻辑视图就是对象模型。逻辑视图重点在于功能，功能包括可见的业务功能，也包括不可见的系统功能（如日志、权限、事务等）。同时更重要的是确立逻辑分层、模块划分和模块之间的依赖关系。

- **开发视图**

描述软件在开发环境下的静态组织。开发视图关注程序包，应用的统一框架，引用的类库、SDK 和中间件，以及工程和包的划分规则等，规范、约束开发环境的结构。

- **过程视图**

描述系统的并发和同步设计。过程视图聚焦在进程、线程等运行时概念，以及相关的并发、同步、通信等问题。如果系统不需要考虑这些方面，本视图可以省略。

- **物理视图**

也叫部署视图，描述软件如何映射到硬件，反映系统在分布/部署上的设计。物理视图需要解决最终如何安装和部署到服务器，以及网络分布的问题。

3.1.2.1 场景视图

杨康在欧阳锋的公司调查了两周，整理了一份调查报告，公司主要有两类角色：老板和员工，客户关系管理的业务场景大致是这样——如图 3-2 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

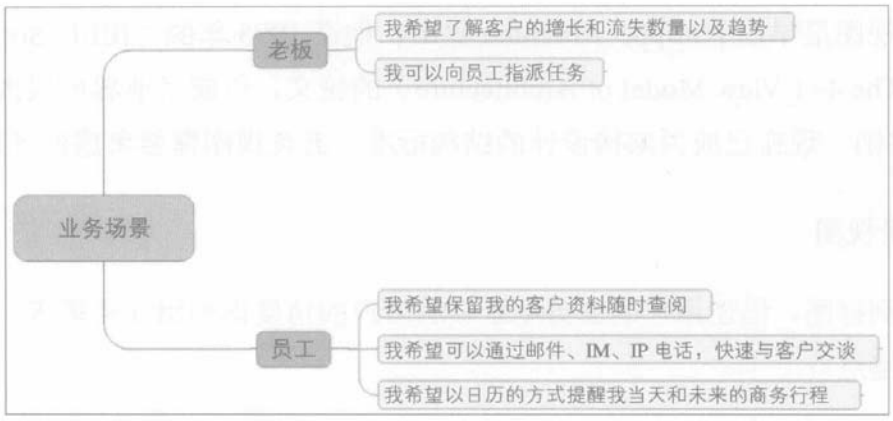


图 3-2　业务场景

- 作为老板，我希望了解客户的增长和流失数量以及趋势；
- 作为员工，我希望保留我的客户资料随时查阅，并且通过邮件、IM、IP 电话快速与客户交谈；
- 作为员工，我希望以日历的方式提醒我当天和未来的商务行程。

……

场景是基于角色来识别的，但不少小公司角色定义比较模糊，角色兼顾的情形比较常见，比如老板有时候也要充当员工，所以郭靖先抛开角色的差异，抽象核心的业务逻辑。当然最终提供给用户的 UI 界面必须可以还原不同的用户场景，而实现的方式类似“我的工作台”。

CRM 的主要用例图如图 3-3 所示。

CRM 的主要用例包括：

（1）管理客户：管理客户资料、联系人等，包括增、删、改、查；

（2）管理行程：管理如拜访客户、合同谈判、签订合同等商业行程的安排；

（3）管理订单：管理与客户签订的订单。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

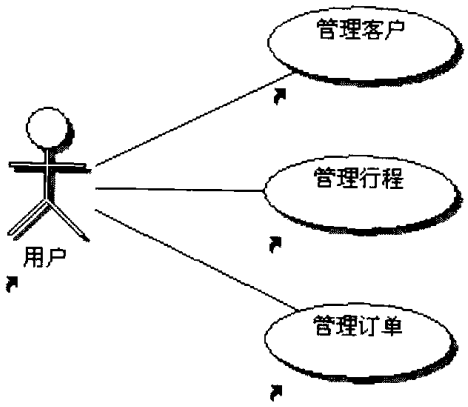


图 3-3　用例图

3.1.2.2　逻辑视图

1. 模块划分与依赖关系

为了更好地分析和表达系统的逻辑结构，区分业务边界，郭靖又根据业务场景将 CRM 系统拆分成【客户】模块、【行程】模块、【订单】模块和【报表】模块 4 大模块（见图 3-4）。

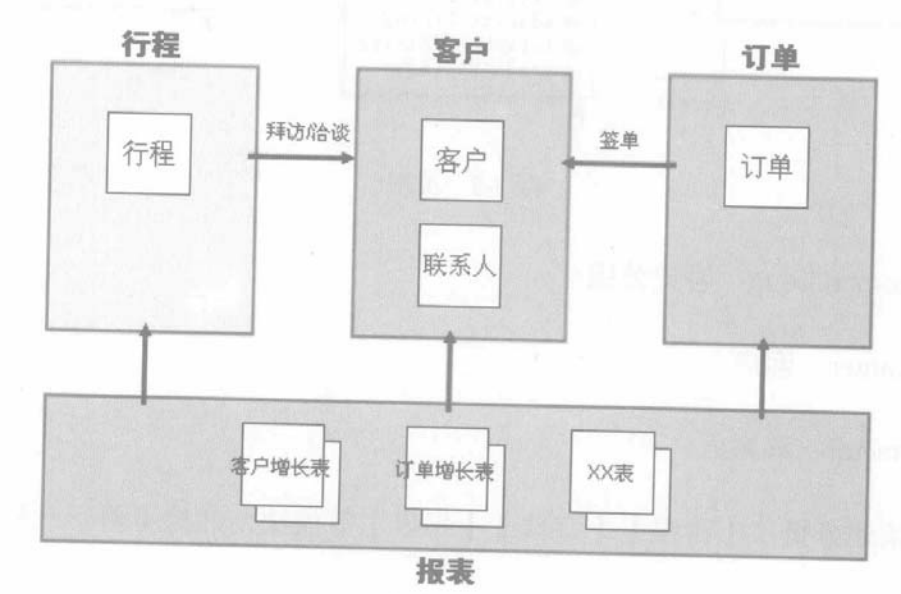


图 3-4　模块结构图

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 模块说明：

模　块	缩　写	功能描述
客户	Customer	管理包括：客户资料、联系人资料
行程	Schedule	管理商务行程
订单	Order	管理用户订单
报表	Report	反映客户增长和流失数据趋势、订单金额和数量增长趋势等

2. 业务实体

分析过需求后不难发现，客户分组、客户、联系人是【客户】模块主要的业务对象，它们的属性和相互关系也很明确，郭靖立刻用 UML Class 图画了下来，如图 3-5 所示。

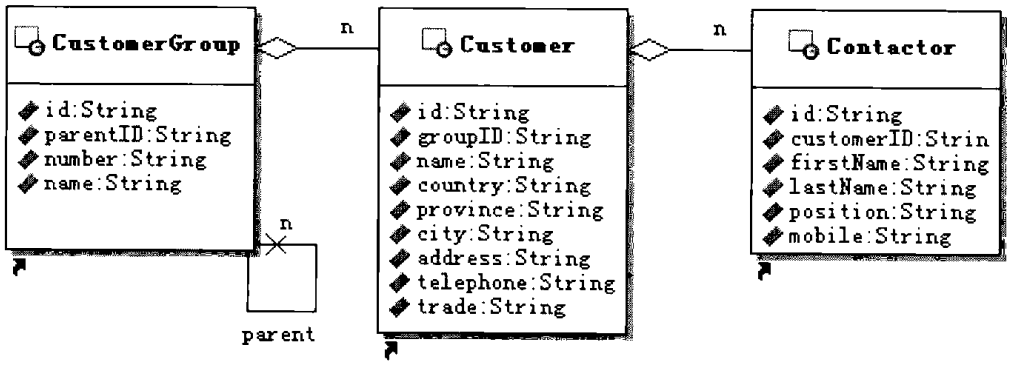


图 3-5　类图

- CustomerGroup：客户分组
- Customer：客户
- Contactor：联系人

同时郭靖还分析了【行程】、【订单】、【报表】等模块，此处不再一一列举。

3.1.2.3　开发视图

前面说到开发视图是用于描述开发环境下的静态组织的，郭靖打算从开发环境、技

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

术架构、分层策略和目录结构 4 个方面加以阐述。

1. 开发环境

对开发环境的说明，需要明确开发的语言、数据库类型、应用服务器类型，以及其他软、硬件要求。

- 开发语言：Java、JavaScript、Html
- 数据库：MySQL 5.0
- 应用服务器：Apache + JBOSS
- 其他软件：Ant、JUnit 等

2. 技术框架

开源框架是现在技术应用的热点，经过数年的“框架大战”，对应表现、业务、持久三层的各种框架各自找到了自己应有的位置。“Struts+Spring+Hibernate”已成为 Java 开发的主流体系。在这个体系中，它的地位在短期内是难以撼动的。郭靖当然不会不知道，其实他对于开源框架早已驾轻就熟，这次他仍旧搬出了“Struts+Spring+Hibernate”的三驾马车式框架结构（见图 3-6）。

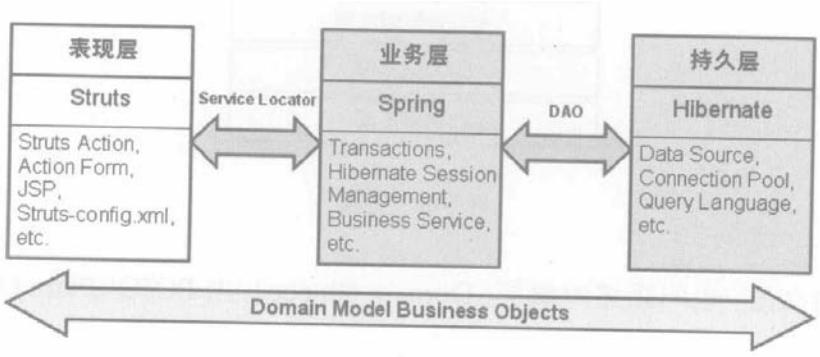


图 3-6　Struts+Spring+Hibernate 框架结构

之前这 3 种框架各自在发展，但如何整合起来发挥最大的能量，形成一个轻量级的框架集呢？郭靖觉得很有必要在开发视图中说明清楚（见图 3-7）。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

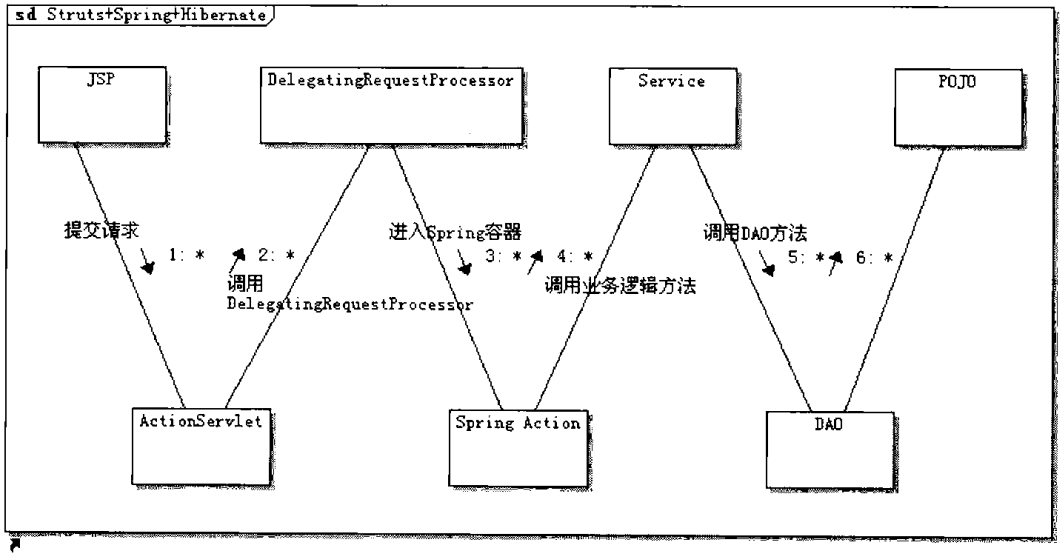


图 3-7　调用关系

3. 分层策略

分层是架构的基础，不管是经典的 J2EE 架构，还是轻量级的 J2EE 架构，一般大致分为 5 层（见图 3-8）。

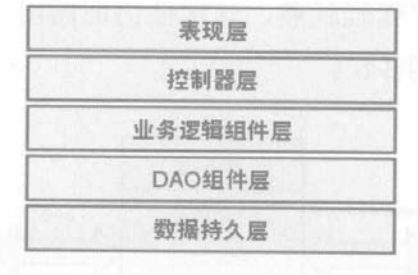


图 3-8　分层

- 数据持久层：也叫领域对象层（Domain Object），由 POJO（Plain Old Java Object）组成。
- DAO 组件层：由 DAO（Data Access Object）组件组成，一般封装了对数据库的 CRUD 原子操作。
- 业务逻辑组件层：一般由 Service 对象组成，实现系统所需要的业务逻辑处理。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 控制器层：控制器用于拦截用户请求，调用业务逻辑组件，根据处理结果转发到不同的表现层组件。
- 表现层：一般由 JSP 组成，负责接收用户请求，反馈处理结果。

4. 目录结构

目录结构与分层策略和打包部署直接相关，根据以上 5 层，郭靖制定了目录结构规则（见图 3-9）。

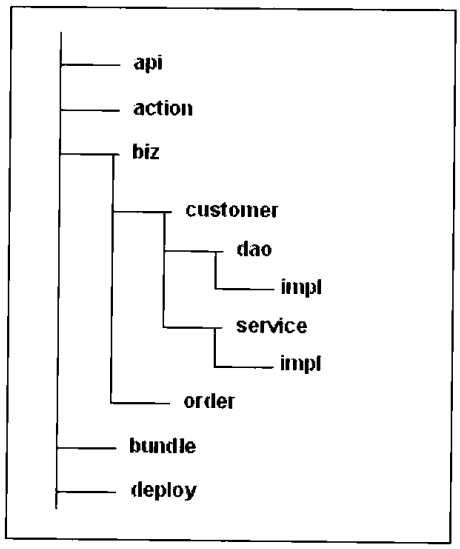


图 3-9　目录结构

- 一级目录包括：
 - ✧ api：存放 service 的接口定义源文件；
 - ✧ action：存放 action 的源文件和配置文件；
 - ✧ biz：存放各模块的业务逻辑组件和 DAO 组件；
 - ✧ bundle：存放 JSP 和 HTML 文件；
 - ✧ deploy：存放构建后待部署的 jar 文件。
- biz 目录下根据模块再分为 dao 和 service 目录，以及它们的实现类目录 impl。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3.1.2.4 过程视图

“4+1”视图本身是可以裁减的，一般而言，“场景视图”和“逻辑视图”必不可少，其他视图依场景而定是否需要，郭靖认为当前的系统暂无并发或同步等问题，所以将过程视图忽略。

3.1.2.5 物理视图

系统部署时将 Web 服务器、应用服务器和 DB 服务器分离，以保证业务繁忙时快速的响应性能（见图 3-10）。

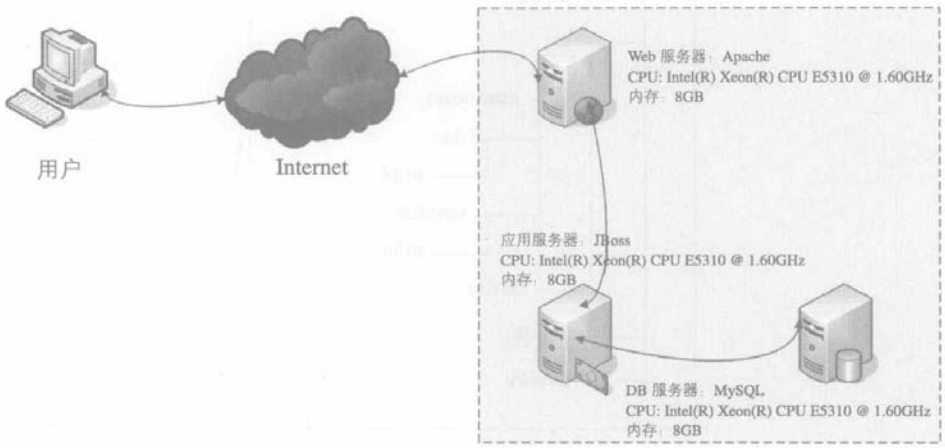


图 3-10 部署图

3.1.3 设备托管

经过一个月左右的设计、开发、测试，这套 CRM 系统基本成型。

郭靖：这两个礼拜我们去了三趟南方，如果项目上线，系统维护肯定少不了，那我们岂不是要经常两头跑，这怎么吃得消……

杨康：我也感觉到了，有没有更好的服务方式呢？

郭靖：我们可以效仿 ASP（Application Service Provider）的做法，将设备托管在我

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

们这里。因为这是一套基于 Internet 的 B/S 系统，欧阳锋的公司只要可以上网，即可登录使用这套系统，而且以后系统的维护/升级、Bug 修复就方便多了。

杨康：不错，这完全是一套双赢的方案，我们在异地维护，欧阳锋也可节省一笔开支，不用再报销我们的差旅费。交给我好了，我去跟他谈。

经过杨康一番摆事实、讲道理，欧阳锋同意了**设备托管**的系统部署方案，一套符合成熟度模型 **Level 1**（定制开发）特征的 SaaS 系统诞生了。而且上线之后欧阳锋比较满意，因为他有时间去打高尔夫了。

3.2 第二阶段：做产品



3.2.1 设备共享

转眼三个月过去了，这天是元旦后的第一个周六，杨康和郭靖正在学生公寓玩网络游戏“魔兽争霸”，这时手机响了。

欧阳锋：杨康，前几天梅超风来我这洽谈业务，正巧看到了你们做的 CRM 系统，她很感兴趣！

（杨康心中窃喜，没想到这么快）

杨康：哦，是吗？

欧阳锋：她也很想装一套，但是她公司的业务与我的有所不同，她有自己的业务员，也有外地加盟的代理商，她希望业务员之间的客户资料保密，业务员只看到自己的客户，你想想办法吧……

接完电话，杨康回头冲郭靖兴奋地大叫。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

杨康：兄弟，我们又有客户了，哈哈。

郭靖：说来听听……

郭靖：这是个新需求！但好办，控制客户资料的数据访问权限，我们在配置管理工具 SVN 中单独创建一个分支，单独开发一个版本，并单独部署一套系统。

杨康：那服务器呢？要不要再买一套？

郭靖：暂时不用，我分析了现有设备的负荷，如果梅超风公司的使用规模跟欧阳锋相类似，完全可以搭建在现有设备上！

杨康：那我们可以省下一笔不小的费用啊，哈哈。

郭靖：这就是**设备共享**的好处！

为了“隔离客户资料”的需求，需要在“客户”Service 组件中添加一个方法：getCustomerListByCreator()（见图 3-11）。

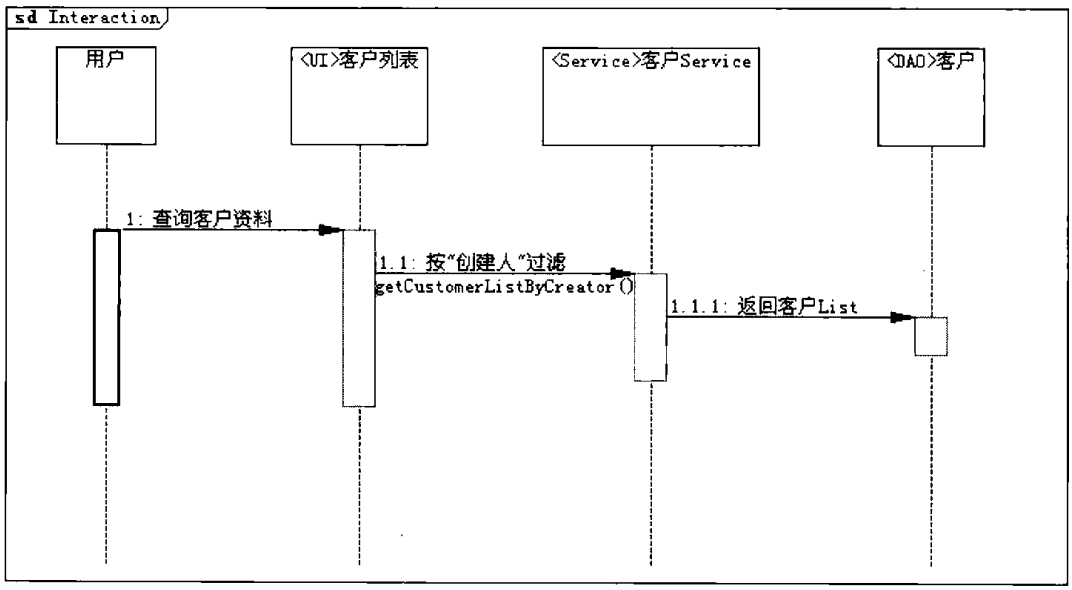


图 3-11　时序图

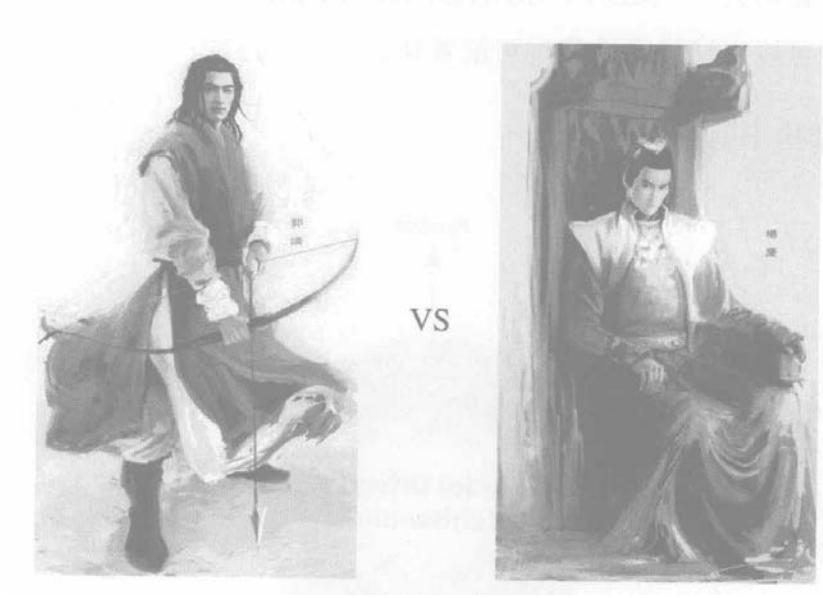
如此这般之后，梅超风公司的 CRM 系统也上线了，虽然其后还有一些很个性化的

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

需求，但原有设计和逻辑基本可以满足。当然美中不足的是从此版本的分支多了起来，杨康、郭靖需要同时响应两家客户的需求变更，**版本管理日益复杂**，这对于一帮学生而言，无论时间和资金都是一大挑战。

3.2.2 创业之旅

话分两头，郭靖和杨康毕业了！不走寻常路，由于大学期间积累了不少项目经验，郭靖和杨康打算成立自己的公司开始创业之旅。公司的名字很酷：**神雕科技有限公司**，杨康任 CEO，郭靖任 CTO，学弟欧阳克也在公司实习。



成立了公司可不一样，必然要有规模和成本意识，在公司成立前后的那段日子里，杨康和郭靖时常一起畅想公司的未来与发展。

杨康：项目面向单一客户，而产品则面向大众或者行业，通用性更强，成为成熟的解决方案后成本更低。

郭靖：如果我们沉淀以往所做项目的用户需求，推出系统化、可配置的 CRM 产品，这样再有客户来，只要简单地安装即可。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

杨康：我们已有欧阳锋和梅超风两家公司的需求，我最近又在谈托雷、丘处机和段皇爷三个项目，抽取共性需求应该并不难。

郭靖和杨康都意识到应该尽快从项目转型成产品，于是，第一代产品化的 CRM 系统开始酝酿……

3.2.3 可配置化

既然是产品，必然可以兼顾多方的共性和个性化需求，郭靖想到了之前在老师洪七公那里学来的一套功夫——MDA（Model Driven Architecture，模型驱动架构），MDA 利用元数据建模，可以方便灵活地实现可配置化。

- MDA 架构（见图 3-12）

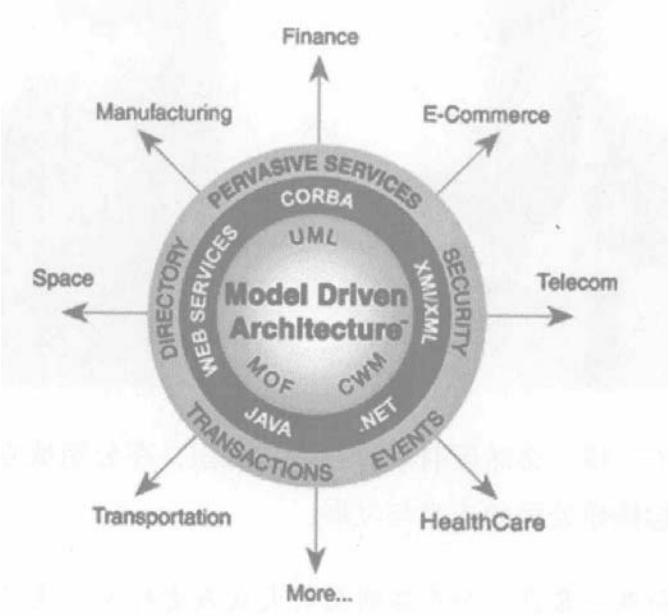


图 3-12 MDA 架构图

- MDA 四层模型（见图 3-13）

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

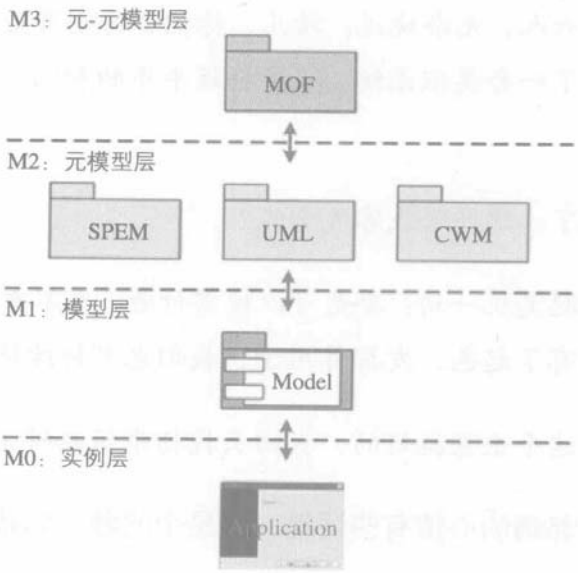
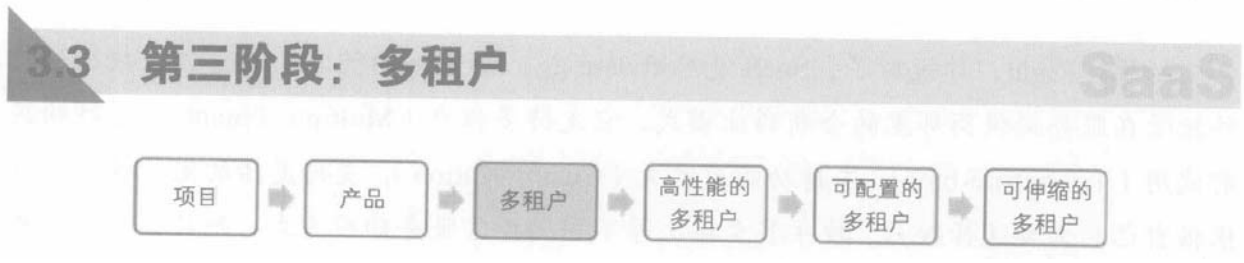


图 3-13　MDA 四层模型

（有关“可配置性”的更多内容请见第 5 章：Multi-Tenant 应用的可配置性。）

当然杨康的市场能力还真不是吹的，产品推出不久，因为强大的可配置灵活性，杨康发展了超过 50 家客户。通过产品可配置化改造，杨康、郭靖的 CRM 系统进化到了 SaaS 成熟度模型的 **Level 2**（可配置）阶段。



3.3.1　成长的烦恼

看着越来越多的客户好了起来，郭靖不免想起了自己幼时的恩人——江南七怪，当年多亏了他们照顾。现在的“**江南贸易公司**”处境并不好，何不让他们用 CRM 呢?

郭靖：柯伯伯，我们公司的 CRM 系统挺实用的，我把它送给你们，希望助你们一臂之力!

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

柯镇恶看了看其他六人，无奈地说：靖儿，你们公司刚开业不多久，开支又大，我听隔壁老板说他当初上了一套类似系统，花了将近半年的利润。免费白送不大好，不合适、不合适啊……

郭靖：柯伯伯言重了，您还跟我客气啥呢？

一旁的妙手书生朱聪灵机一动：要是可以按需付费，岂不更好？一来我们不会一开始投入过大，二来生意有了起色，发挥作用了，我们也可持续使用。

郭靖：按需付费？这个主意挺好的，我回头找杨康研究研究……

见过江南七怪后，郭靖的心情有些沉重，这是个问题，郭靖拉来了杨康商量。

郭靖：表面上看公司的客户已经几十家，但是不是还有大量的潜在客户我们没有挖掘，这些客户规模较小、实力较弱，可他们同样有强烈的管理需求，我们为什么不把门槛降低、规模做大呢？

杨康：中国有超过 2000 万的中小企业，他们才是我们未来的蓝海！

郭靖：听说现在有一种 SaaS 多租户模式，跟我们想象的场景很相似，但我没弄明白是怎么回事……

杨康：Right，你说对了，SaaS 是 Software as a Service 的缩写，是一种将数据和软件托管在服务提供商那里的全新商业模式，它支持多租户（Multiple Tenant），支持购买前试用（Try-before-buy），支持功能自定义（Customization），支持灵活的定价策略，可根据自己的需要选择按月、按年甚至按次等不同的按需服务购买方式，而且客户不需要任何别的其他投入。

杨康：最近国外有家做 CRM 的公司【Salesforce】非常火，它就是典型的 SaaS 应用。

郭靖：那我们 Google 一下……

郭靖迫不及待地打开了电脑——

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Salesforce.com 是全球按需 CRM 解决方案的领导者，当前全球有 43,600 多家公司和 646,000 名注册用户正使用 Salesforce 的强大功能分享客户信息，以及开发具有更高收益的客户关系。Salesforce.com 如何能提供比别人更多的成功机会？方法很简单：

强大的功能：当今业界技术最先进的产品——第 20 代产品具有 1 000 多种功能，因此具有统领全球业务的能力；

灵活的定制：这是业界灵活度最高的 CRM 解决方案，独有的自定义选项卡和全新设计的 Customforce 令自定义灵活度显著提高，用户可深度扩展，因此能满足各种规模的企业的需求；

最佳的用户体验：方便易用，简洁的界面一目了然；

多语言支持：支持 14 种语言；

按需应用、按需付费；

快速实施：多数公司在 30 天之内把 Salesforce.com 成功融合于企业运转之中；

快速回报：通常在实施后的几个月之内，客户即可获得可观的回报；

没有安装费：软件托管于 Salesforce.com 公司的强大数据中心，用户只需登录即可使用，免除软硬件购买、安装、调试过程；

高度安全：系统和数据处于层层保护之中；

免费版本更新；

免费用户支持。

郭靖：简直太妙了，这种商业模式肯定火！

杨康：为什么不呢，我马上起草报告改变公司今年的战略——进军 SaaS！

3.3.2 如何转化成 SaaS 多租户模式

SaaS 虽好，但怎么样才能转化成 SaaS 多租户应用？从技术的角度分析需要改造哪

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

些地方？带着这些疑问，郭靖去找他的导师洪七公。

洪七公：SaaS 现在已成为一股潮流，它将颠覆传统的软件交付方式，你们的产品打算走 SaaS 路线，我非常支持。

洪七公：其实从架构层面来分析，SaaS 区别于传统技术的重要差别就是 **Multi-Tenant** 模式……

郭靖：Multi-Tenant？多租户？

洪七公：是的，多租户就是说多个租户共用一个实例，租户的数据既有隔离又有共享，说到底就是如何解决数据存储的问题。

郭靖：哦，隔离？共享？

洪七公：现在 SaaS Multi-Tenant 在数据存储上存在三种主要的方案，分别是——

3.3.2.1 独立数据库

这是第一种方案，即一个 Tenant 一个 Database（见图 3-14），这种方案的用户数据隔离级别最高，安全性最好，但成本也高。

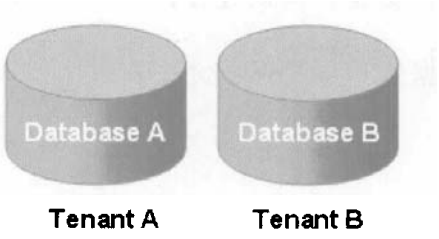


图 3-14 方案一

- 优点：
 - ✧ 为不同的租户提供独立的数据库，有助于简化数据模型的扩展设计，满足不同租户的独特需求；
 - ✧ 如果出现故障，恢复数据比较简单。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

■ 缺点：

✧ 增大了数据库的安装数量，随之带来维护成本和购置成本的增加。

这种方案与传统的一个客户、一套数据、一套部署类似，差别只在于软件统一部署在运营商那里。如果面对的是银行、医院等需要非常高数据隔离级别的租户，可以选择这种模式，提高租用的定价。如果定价较低，产品走低价路线，这种方案一般对运营商来说是无法承受的。

3.3.2.2 共享数据库，隔离数据架构

这是第二种方案，即多个或所有租户共享 Database，但一个 Tenant 一个 Schema（见图 3-15）。

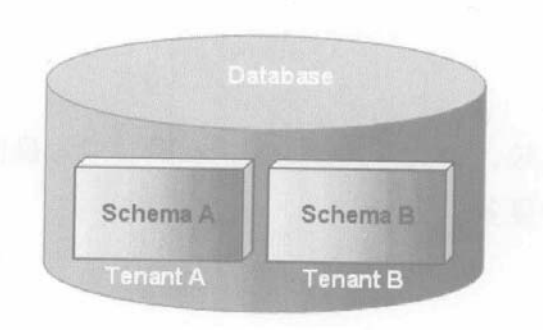


图 3-15 方案二

■ 优点：

✧ 为安全性要求较高的租户提供了一定程度的逻辑数据隔离，并不是完全隔离；

✧ 每个数据库可以支持更多的租户数量。

■ 缺点：

✧ 如果出现故障，数据恢复比较困难，因为恢复数据库将牵扯到其他租户的数据；

✧ 如果需要跨租户统计数据，存在一定困难。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3.3.2.3 共享数据库，共享数据架构

这是第三种方案，即租户共享同一个 Database、同一个 Schema，但在表中通过 TenantID 区分租户的数据（见图 3-16）。这是共享程度最高、隔离级别最低的模式。

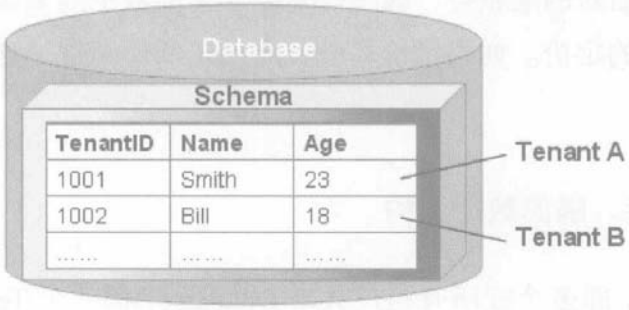


图 3-16 方案三

- 优点：
 - ✧ 三种方案比较，第三种方案的维护和购置成本最低，允许每个数据库支持的租户数量最多。
- 缺点：
 - ✧ 隔离级别最低，安全性最低，需要在设计开发时加大对安全的开发量；
 - ✧ 数据备份和恢复最困难，需要逐表逐条备份和还原。

如果希望以最少的服务器为最多的租户提供服务，并且租户接受以牺牲隔离级别换取降低成本，这种方案最适合。

3.3.2.4 三种方案比较

洪七公：从隔离和共享两个相反的方向比较，依次是方案一、方案二和方案三，如何选择取决于产品的定价策略和租户对数据安全的接受程度。

三种方案比较见图 3-17。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



模式	隔离级别	共享级别	安全性	成本
独立数据库	高	低	高	高
共享数据库，隔离数据架构	中	中	中	中
共享数据库，共享数据架构	低	高	低	低

图 3-17　方案比较

郭靖：老师，我明白了。我们的 CRM 系统未来将以中低端市场为主，我们决定采用第三种方案，只要做好数据隔离就好了。

洪七公：千万不可掉以轻心，SaaS 下的安全性设计很重要。一般常见的安全性设计分为两类：系统级和程序级。

- 系统级：
 - ✧ 使用 HTTPS 协议以 SSL（Security Socket Layer）交换数据，增强通信安全；
 - ✧ 通过数字签名防止传输过程篡改；
 - ✧ 对用户身份识别的 UserToken 使用 DES 算法数据加密；
 - ✧ 业务数据定时自动备份。
- 程序级：
 - ✧ 完整的权限配置，包括功能权限和数据权限；
 - ✧ 客户端输入校验，防止 JS 攻击、XSS 攻击、SQL 注入等；
 - ✧ 辅助安全设计，比如密码控件、图片验证码、手机确认码等。

郭靖：我知道该怎么去做了，谢谢老师指点！

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3.3.3 SaaS 多租户设计

决定采用“共享数据库，共享数据架构”方案后，郭靖着手开始改造。改造成 SaaS 多租户的重点在于租户管理和数据隔离。

- 租户管理：包括注册、订购、计费等；
- 数据隔离：变更表结构，增加 TENANT_ID 字段。

3.3.3.1 租户管理

既然 SaaS 是敞开大门做生意，允许任意客户可以通过互联网随时随地开通、订购和使用产品，那么 SaaS 的租户管理就显得格外重要。

- 注册（见图 3-18）

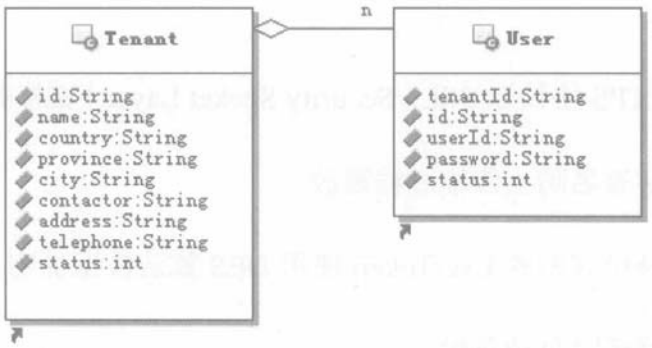


图 3-18 类图_注册

- Tenant：租户；
- User：租户下的用户账号，一个租户下可以有多个用户账号。

Tenant 的 status 属性用于定义租户的当前状态：待审核、已审核、启用、禁用、取消等。登录时必须判断租户的状态，只有处于“启用”状态的租户才能登录进入系统。

- 订购（见图 3-19）

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

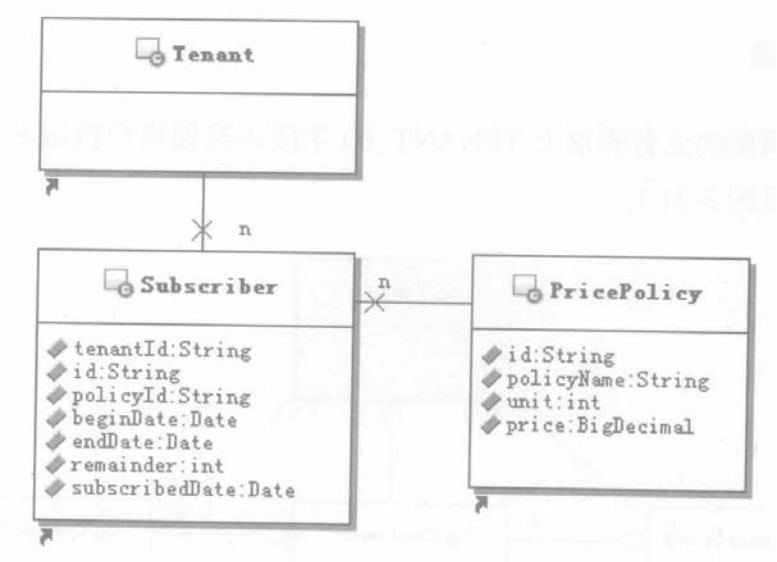


图 3-19　类图_订购

- PricePolicy：价格策略，支持按时间（月、季、年）、按次数计价，以 unit 表示；
- Subsciber：订购记录，记录租户选择了何种价格策略，以及服务期限或服务次数。

■ 计费（见图 3-20）

- Journal：流水账，记录租户使用服务的日志，用于生成账单和报表。

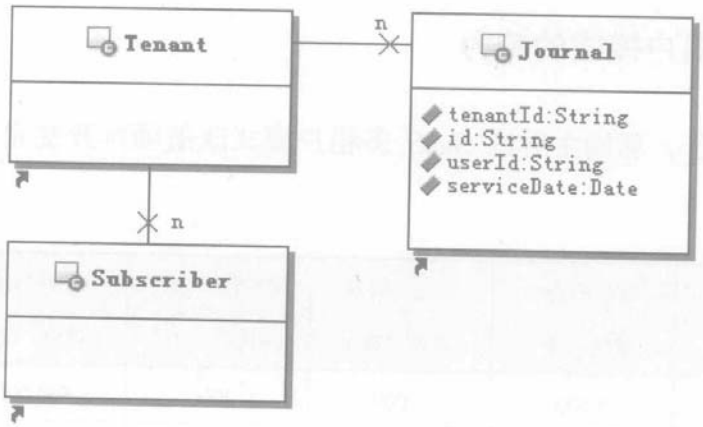


图 3-20　类图_计费

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3.3.3.2 数据隔离

为每个需要隔离的业务表加上 TENANT_ID 字段以实现租户数据之间的隔离，这是最通常的做法（见图 3-21）。

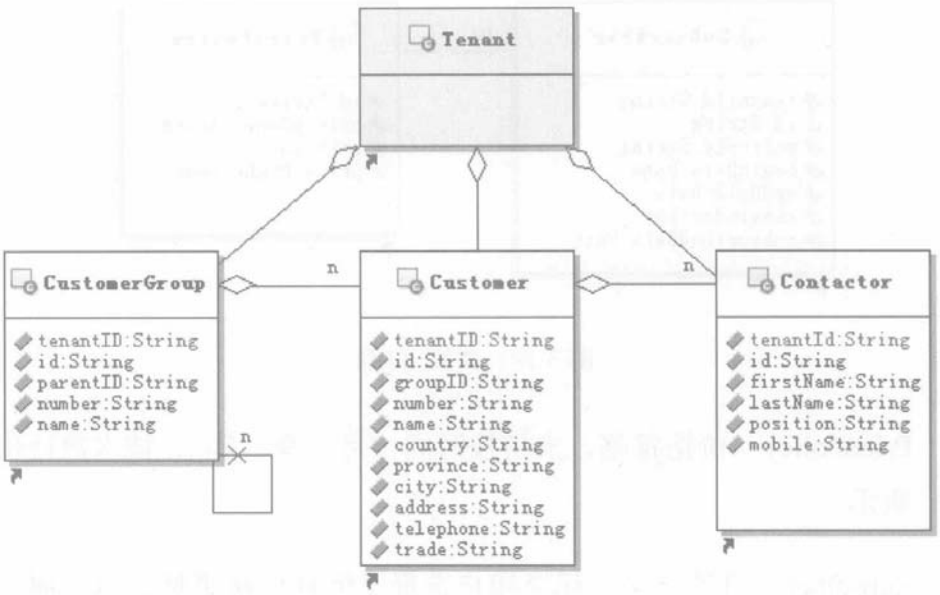


图 3-21 类图_客户管理

对比前后的 DB 设计，多出的 TENANT_ID 属性是 SaaS 多租户模式相对于传统模式最大的不同点。

3.3.4 SaaS 多租户模式的威力

又三个月过去了，郭靖主导的 SaaS 多租户模式改造项目开发完毕，杨康忙不迭地算了一笔账：

项 目	每客户收入（单位：元）	每客户成本（单位：元）	每客户利润（单位：元）	预计客户数（单位：个）	预计利润（单位：元）	人均支持客户数（单位：个）
改造前	8,000	5,800	2,200	300	660,000	10
改造后	500	220	280	50,000	14,000,000	150

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

很明显，因为改造成 SaaS 多租户后单一成本极大降低，价格亦大幅降低，自然可吸纳大量的付费用户。对比前后的利润增长 20 余倍，技术支持人员人均可支持的客户数也增加 10 多倍，效果十分显著。看过这份表格后，郭靖和杨康不由轻松地畅想起了未来——

郭靖：这回我可以腾出更多的人力去提升性能和可伸缩性了，Yeah!

杨康：这回我可以多招些 Sales 去拓展渠道和市场了，哈哈。

经过数据隔离相关改进后，杨康、郭靖的 CRM 系统更进一步，已经实现了 SaaS 成熟度模型 **Level 3**（多租户）阶段的主要特征。

3.4 小结

本章以 CRM 系统为例，以软件交付形态由做项目到做产品，再到多租户的转变，反映了 SaaS 成熟度模型从 Level 1，到 Level 2，再到 Level 3 的演进过程。核心内容包括：

- 通过“4+1”视图设计架构；
- SaaS 多租户模式下三种数据隔离方案；
- SaaS 多租户设计。

从本章内容中我们可以看出 SaaS 成熟度模型不同等级的不同特点：

- Level 1：设备托管；
- Level 2：设备共享、可配置化；
- Level 3：多租户、数据隔离、高性能。

如果希望达到 SaaS 成熟度模型的 Level 3 级要求，最核心需要解决的是数据隔离的问题。后续章节将就如何达到 Level 4 级要求进一步展开。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第4章 高性能的 Multi-Tenant 最佳实践

4.1 数据库层性能优化 …… 75
4.1.1 建立合适的索引 …… 76
4.1.2 消除大数据表连接 …… 77
4.1.3 避免复杂 SQL …… 78
4.2 应用层性能优化 …… 78
4.2.1 Cache …… 79
4.2.2 统计和报表计算 …… 82
4.2.3 基于 Tenant 的索引搜索 …… 84
4.2.4 异步操作 …… 85
4.3 Web 层性能优化 …… 87
4.3.1 Web 开发的性能优化策略 …… 88
4.3.2 Http 服务器的性能优化策略 …… 91
4.4 性能监控 …… 93
4.5 优化的效果 …… 95
4.6 小结 …… 95

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

杨康的销售和市场拓展能力还真不是盖的。侠客录 CRM 的 SaaS 版本一经上线，迅速吸引了大批的中小企业客户。企业只须每人每月支付 50 元人民币，即可享受强大的客户关系管理功能，而无须自己架设服务器和网络设备，还省掉了 IT 维护的人工成本。侠客录 CRM 还提供长达 3 个月的免费试用期，这实实在在给老板们吃了定心丸，降低了软件使用风险，还降低了使用成本。一时间，大量的企业开始尝试侠客录 CRM 系统，特别是中小企业的老板们纷纷踊跃试用和订购。

不过很快郭靖的压力就上来了。短短 1 个月时间已经有 500 多个客户开始试用侠客录 CRM，高峰期的同时在线用户数也在 100 以上。这造成了应用服务器和数据库服务器的压力非常大，高峰时用户的平均响应时间在 10 秒左右都是很常见的了。很快杨康就找上了郭靖。

杨康：最近我推广软件的时候，怎么老有客户跟我抱怨我们的软件很慢呀？

郭靖：是的，用户太多了，服务器的压力太大，响应不过来。

杨康：那有什么办法没有呀？

郭靖：有是有，不过就是价格不是我们这种小公司能够承受的，买几台小型机可是要几百万呀。

杨康：那肯定是不可能的了，我们再买几台 PC Server 不行吗？

郭靖：有难度呀，我们之前的软件是设计为单个公司使用的，并没有太多考虑性能和扩展性方面的问题，要做这个改造挺复杂的。

杨康：没问题的，我相信没有什么技术问题可以难倒我们的技术天才的。我三个月估计可以带来 2000 左右的客户，你在后方要给我们顶住！

杨康撂下一个高帽子，就又去会见客户了。办法只能郭靖一个人慢慢地想……

4.1 数据库层性能优化

传统企业应用改造成 Multi-Tenant 架构时，或者传统企业软件开发者开发

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Multi-Tenant 架构应用时，首先可能面临的性能问题来自于数据层。就像之前郭靖的侠客录 CRM 系统一样，每个数据库面向的是单个企业客户，很多时候并不需要过多考虑数据层的访问压力。毕竟一个企业，尤其是他们所面向的中小企业客户，无论是用户量还是数据量都不会太大。但是改造成 Multi-Tenant 架构之后，成百上千的企业共用同一个应用实例和数据库实例，系统的并发压力和数据量都会成百倍地增加。这种情况下，如果继续按照传统企业软件开发的思维方式，数据层的压力问题就会显而易见。这就是为什么郭靖的 CRM 系统响应变慢的主要原因。

4.1.1 建立合适的索引

建立合适的索引对于基于数据库的应用都非常重要。而相对于传统企业软件来说，为 Multi-Tenant 架构软件中的数据建立合适的索引并有效利用索引，更是至关重要。毕竟对于 Multi-Tenant 架构软件，其数据量和并发度都会是传统企业软件的几百或几千倍以上。这时候，如果一个数据库操作不能有效利用索引，其不得不面临对一个庞大数据表的扫描，从而对于整个系统的数据库的综合性能造成极大的影响。

对于 Multi-Tenant 架构的软件，除了根据软件之前的业务建立的索引之外，在进行 Multi-Tenant 改造过程中，基本上在大部分业务数据表中都增加了 TENANT_ID 字段。因此，应用需要做如下调整：

- 在原有的索引上，都增加一个 TENANT_ID 字段。由于 TENANT_ID 的选择性较强，一般 TENANT_ID 字段需要作为联合索引的第一个字段。
- 所有的业务查询操作，都增加 TENANT_ID 字段进行过滤查询。

关于具体如何建立合适的索引，可能更多的是 DBA 关心的专业问题。这里简要描述一下：

- 索引应该创建在条件（where）、排序（order by）、分组（group by）等操作所涉及的列上。
- 索引应该有较强的选择性，即应尽可能建立在重复数据少的数据列中。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 如果多个条件经常需要组合起来查询，应合理使用联合索引。
- 一次查询中只能使用一个索引，可使用相应的分析工具分析索引效果。
- 索引不是越多越好，过多的索引可能导致 CUD（新增、修改、删除）的性能降低，并且占用更多的空间。

因此，对于郭靖的侠客录 CRM 系统，首先要做的就是在原有的索引上都加上 TENANT_ID 字段，并且在相应的查询中增加 TENANT_ID 字段作为过滤条件。然后找出耗时较长的 SQL，逐一分析其索引是否合理，是否有效利用索引等。经过索引的优化，整个系统的性能很快就能产生飞跃，这也是最简单的办法。

建立了合适的索引之后，仍然还有一些数据库查询不能有效利用索引，例如包含 like 等模糊查询。对于这一点，在互联网应用中继续使用模糊查询显然是不太现实的，要实现模糊查询，更适合的方式是搜索引擎，只是应用的复杂度会增大不少，这一点在下面的搜索引擎一节也会详细说明。

4.1.2 消除大数据表连接

但是并不是所有的耗时长的 SQL 问题都是由于索引引起的。索引优化完成之后，就会发现太多的大数据量表连接操作占据了耗时 SQL 统计中的大部分。在传统企业软件模式下，一个数据库只供一个企业所使用，而一个企业的订单数量大部分情况下都是很有限的。尤其对于侠客录 CRM 所面向的客户群而言，订单数量超过几万个的客户应该也是很少的。不过在 SaaS 模式下，由于有了数百个租户，订单表的记录数量达到百万级就是很轻易的事情了。此时，对于单个表的查询操作，由于采用了适当的索引策略，一般性能问题还不大。而对于多表关联的情况，原来的在软件模式下运行得很正常的 SQL，到了 SaaS 模式下性能问题就显得很突出。

要避免由于大数据表连接引起的性能问题并不容易，不同的场景下需要采取不同的策略解决。表 4-1 列举了常见场景下的几种解决方案。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 4-1　消除表连接的几种解决方案

解决方案	适用场景	示例场景
冗余存储关联字段	业务需求上可以接受冗余导致的不一致，或者冗余数据可以很容易被同步更新	订单列表查询时，希望查看到订单的客户名称，原本订单上只记录了客户 ID，通过关联客户表查询客户名称
Cache 缓存	变动概率不高，但是对于数据一致性要求较高	用户名（USER_NAME）被很多业务所关联查询，但是也不适用于冗余方案（业务上不允许不一致，并且要保持所有冗余存储的地方同步更新很困难）
直接删除关联字段	不是必须包含的被关联表的字段，可以直接从列表查询中去除	订单列表中的产品型号等非关键字段，其实并不一定要包含在订单列表中
拆分成多次查询	对于单个数据的查询，如果涉及多张关联表，有时分多次查询会比一次复杂的关联查询更为合适	订单表单中需要查询到关联产品的编码、型号等很多字段

通过以上这些措施，我们可以避免相当部分的不必要的大数据表连接。数据库服务器的压力能够显著降低，因此系统性能能够得到显著提升。

4.1.3　避免复杂 SQL

不必要的大数据表连接也消除了之后，继续跟进耗时 SQL 统计，还是会有一些复杂的 SQL 在耗时 SQL 统计中排名靠前，这些复杂的 SQL 很难有统一的解决方案。大部分情况下可能需要 case by case 去处理，不过有一些通用性的处理原则可供参考：一般需要考虑将复杂的 SQL 分解成多个简单的 SQL 执行，将复杂的运算尽量留在应用服务器层。因为一般情况下，应用服务器层的水平扩展更容易实现，而数据库层的扩展则是比较困难的。

4.2　应用层性能优化

应用层不是我们首先提及的性能优化对象，原因是在大部分情况下，应用层更容易

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

通过简单的水平扩展来增强系统处理能力。不过，对于郭靖这样的创业型公司，在用户量还不足够多的情况下，首先考虑增加服务器并不太现实，因为其前期投入成本较大。因此，首先还应该是从技术角度考虑性能优化。应该说应用层的性能优化策略有很多，这里描述了常见的几种被广泛采用的技术方案。

4.2.1 Cache

其实很难说 Cache 就是应用层的优化策略。因为大部分情况下，Cache 所缓存的内容就是数据库中存储的内容。采用 Cache 策略其实也是对数据库层的一种优化，因为其避免了对于数据库的频繁访问。不过，其到底应该被归类于应用层的优化策略还是数据库层的优化策略并不重要，重要的是我们应该在适当的场合下应用 Cache 技术，提升系统性能。

对于互联网应用，Cache 是被广泛采用的技术。早在 2006 年，就曾听说知名的 SNS 网站 Facebook 有多达 400 台以上的 MemCached 服务器，用于缓存各种数据。对于从传统企业应用改造而来的 SaaS 应用，使用适当的 Cache，也是提升性能的一个重要措施。

与互联网应用“读多写少”不太一样的是，SaaS 类应用读写的比例并不悬殊。因此，通常情况下，SaaS 类应用很少会在展现层做 Cache，而更多 Cache 都是针对那些变化相对比较少的业务数据。例如，对于侠客录 CRM 中用户数据、权限角色数据、国家省市地区数据做缓存等。

针对侠客录 CRM 中的用户权限角色数据做缓存，应该是应用 Cache 来大幅提升性能的一个典型场景。侠客录 CRM 中的权限系统结构是一个典型的基于 RBAC（Role-Based Access Control，基于角色的访问控制）模型的设计，其主要涉及的实体关系如图 4-1 所示。

在改造前，每次权限检查时会根据用户、权限、角色的关联表查询到用户对于某项操作是否有权限，每次权限检查都会涉及几次数据库的操作。这种方式在传统的企业应用中实现起来很简单，也没有多大的性能问题。然而，在 SaaS 模式下，这个问题就变得很突出。在多租户的情况下，SaaS 应用的并发度非常高，而几乎每个请求都会涉及权

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

限检查。因此，权限检查对于数据库访问的压力就可想而知。

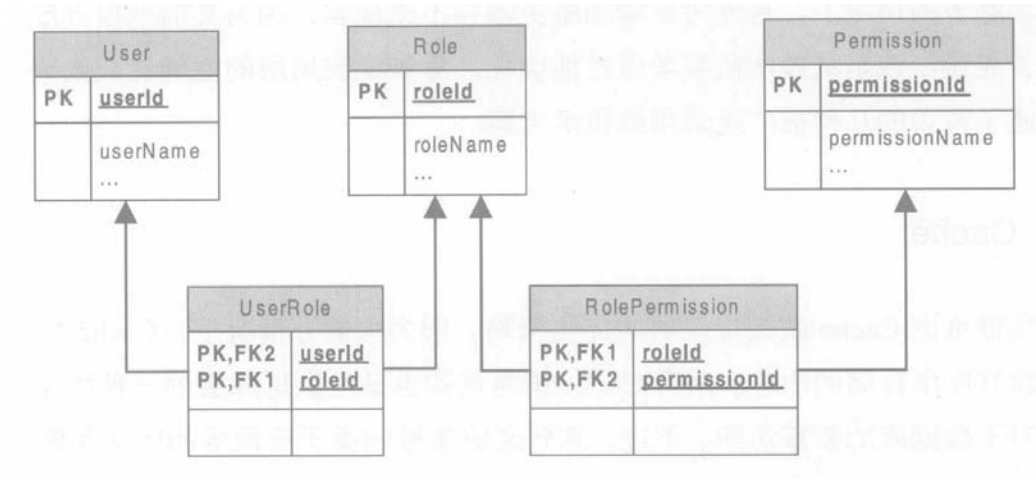


图 4-1　基于 RBAC 的权限系统数据模型

权限数据作为被频繁读取而改动很少的数据，是可以做 Cache 的首选数据。因此，权限系统采用 Cache 技术来提升性能是显而易见的。

但是，就具体采用什么样的 Cache 技术上，可供选择的 Cache 技术有很多，诸如 MemCached、JBoss Cache、OSCache、EHCache 等。其中，MemCached 和 JBoss Cache 应该是两类比较典型性的 Cache，选型比较也将基于这两者。

MemCached 是由 Danga Interactive 开发的高性能、分布式的内存对象缓存系统，用于在动态应用中减少数据库负载，提升访问速度。

通过在内存里维护一个统一的巨大的 hash 表，MemCached 能够用来存储各种格式的数据，包括图像、视频、文件以及数据库检索的结果等。

MemCached 访问速度非常快。MemCached 使用了 libevent（如果可以的话，在 Linux 下使用 epoll）来均衡任何数量的打开链接，使用非阻塞的网络 I/O，对内部对象实现引用计数（因此，针对多样的客户端，对象可以处在多样的状态），使用自己的页块分配器和哈希表，因此虚拟内存不会产生碎片并且虚拟内存分配的时间复杂度可以保证为 O(1)。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

JBoss Cache 是 JBoss 的一个子项目。JBoss Cache 是基于内存的、可复制的、事务性的缓存系统，并且具有持久化状态存储管理能力。JBoss Cache 基于 JGroup 实现 Cache 内容的同步，从而实现分布式 Cache。

表 4-2 MemCached 和 JBoss Cache 的特性比较

	MemCached	JBoss Cache
特性	1. 基于 Client/Server 架构 2. 只有一份数据 Copy，不需要数据同步	基于 JGroup 多播的分布式 Cache
优势	不需要数据同步，避免复杂的多播等技术	Cache 读取基于本地 Memory，性能更高

MemCached 和 JBoss Cache 分别是集中式 Cache 和分布式 Cache 的典型代表。表 4-2 对这两种 Cache 的特性做了一个简单的比较。由于 JBoss Cache 依赖于多播，应用起来可能会有些网络环境的限制。因此，在对性能要求不是特别苛刻的情况下，使用 MemCached 会显得更为简单。于是，郭靖在侠客录 CRM 的权限系统优化中，首先就使用了 MemCached。

在确定了采用 MemCached 之后，就需要确定具体哪些数据在什么时候放入 Cache，什么时候更新。对于权限系统而言，哪个角色具有哪些权限（RolePermission）是相对比较固定的。因此，可以考虑在系统启动时即载入 Cache，在发生变化的时候，更新 Cache。而由于用户具有哪些角色（UserRole）是与用户相关的数据，则应该在用户登录的时候获取用户的角色信息，并放入用户 Session 中。这样大部分情况下验权过程中就不需要涉及任何数据库查询，通过 Session 和 Cache 的访问即可完成，权限系统整体性能可以得到有效提升。

不过，在使用 Cache 过程中还是需要注意，Cache 数据是易失的，并不宜将 Cache 作为一种绝对可靠的数据访问机制。因此，为了保证系统的可靠性，一般情况下应用中还需要增加一层逻辑：当 Cache 中访问不到时，需要访问数据库，并重新将其载入 Cache。

基于数据库的权限验证类图和基于 Cache 的权限验证类图如图 4-2 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

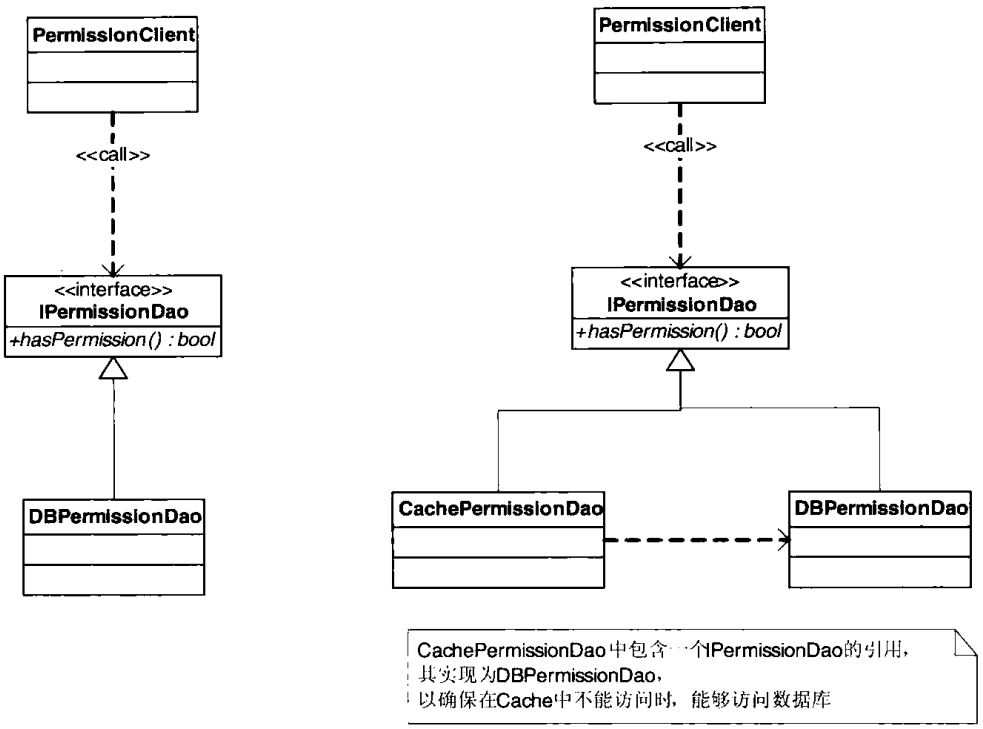


图 4-2　基于数据库的权限验证类图和基于 Cache 的权限验证类图

此外，在上述设计中，用户角色是存放在 Session 中的，如果在用户登录后该用户又被管理员重新分配了角色，则重新分配的角色需要待用户下次登录之后才能生效。不过这个限制一般情况下还是可以接受的。

总结下来，可以使用 Cache 的应用应该具备如下特征：

（1）需要 Cache 的数据应该是读多写少的。

（2）由于 Cache 数据是易失的，因此为了保证可靠性，一般情况下应用中还需要增加一层逻辑：当 Cache 中访问不到时，需要访问数据库，并重新将其载入 Cache。

4.2.2　统计和报表计算

在传统企业软件中，基于实现的简单性考虑，大部分统计和报表都是实时计算的。这种方式在面向单个企业用户的情况下，一般也不会成为太大的问题。考虑到统计和报

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表计算的 SQL 查询大多比较复杂，在 Multi-Tenant 架构下，由于并发量和数据量的增大，此类实时计算的统计和报表也将是造成数据库压力过大的一个主要因素。

例如，图 4-3 是侠客录 CRM 系统的客户首页的一些统计数据。尽管页面不大，但是涉及的统计数据多达数十项。此类的统计计算在 SaaS 模式下继续采用实时计算的方式会对数据库服务器造成巨大的压力，是不太现实的。



图 4-3　侠客录 CRM 的客户首页的统计数据

为了缓解实时计算给数据库带来的巨大压力，统计和报表运算有如下的一些优化策略，不过具体优化策略还是跟应用场景相关：

- 对于统计结果实时性要求不高的（例如大部分的月度报表计算都是针对历史数据的），并且历史数据是不允许修改的情况下（例如财务数据通常有结账的概念），通常可以采用后台任务定时统计的策略。
- 如果历史数据是可以修改的，则相应的后台任务定时统计方案会略有些复杂：对于被修改的历史数据，历史统计结果和报表需要重新运算。
- 对于部分统计结果实时性要求很高的功能，则可以在后台任务定时统计的基础上，再加上增量数据实时统计的结果。这样的一种策略虽然还是有部分实时统计在里面，但是实时统计的数据量就只有部分时间段的增量数据。相对于完全的实时统计运算，这种方式下实时统计的数据量依然会有大幅减少，因此也是一种常见的实时报表实现策略。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

4.2.3 基于 Tenant 的索引搜索

模糊查询是应用软件常见的需求。在传统软件模式下，每个软件实例只为一个企业服务。无论是数据库的数据量，还是访问的并发量，都不会太大，模糊搜索对于数据库的压力大多是可以接受的。在这种情况下，直接采用数据库本身的 like 查询显然是实现模糊搜索的最简单的方案。

在 Multi-Tenant 架构下，无论是数据库的数据量，还是访问的并发量，相对于传统企业软件模式，都有几个数据量级的飞跃。在这种情况下，依然采用数据库本身的 like 查询是万万不会允许的。那这时候需要采取什么样的一些策略来优化性能呢?

最简单的方法当然是不允许模糊查询。虽然是一种偷懒的做法，但是在很多场景下也确实是有效的。对于很多简单的查询并不一定都需要模糊查询，这种情况下，不支持模糊查询也许就是最佳的性能优化方案。并不是任何问题都需要相应的技术方案来解决，很多时候用非技术的手段来解决也许是更快捷高效的方式。这是很多技术人员常常意识不到的。

不过，显然并不是所有的模糊查询都是可以取消的。在很多应用场景下，模糊查询还是必须实现的功能，简单地取消模糊查询还是不行的。这时，最简单和直观的一种想法是利用数据库本身的全文检索功能。目前，主流的数据库都具备全文检索功能，因而直接使用数据库本身的全文检索显然是最简单和直观的做法。侠客录 CRM 在最初处理模糊搜索时也首先考虑使用数据库本身的全文检索功能。

不过，在随后的用户实践中，郭靖很快就会发现数据库本身的全文检索功能并不是模糊查询的最佳选择。因为，其查询结果大多数时候并不能符合用户的需求。这时，搜索引擎的优势就会体现出来。搜索引擎可以更好地为用户提供最可能满足其需求的前 100 条搜索结果，而这在通常情况下可以满足 95%的用户需求。

在开源的搜索引擎中，Lucene 显然是轻量级搜索引擎的最佳选择。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Lucene 是 Apache 软件基金会 jakarta 项目组的一个子项目，是一个开源的全文检索引擎工具包。它不是一个完整的全文检索引擎，而是一个全文检索引擎的架构，提供了完整的查询引擎和索引引擎、部分文本分析引擎。Lucene 的目的是为软件开发人员提供一个简单易用的工具包，以方便在目标系统中实现全文检索的功能，或者是以此为基础建立起完整的全文检索引擎。

Lucene 的原作者是 Doug Cutting，是一位资深的全文索引/检索专家，曾经是 V-Twin 搜索引擎的主要开发者，后在 Excite 担任高级系统架构设计师，目前从事于一些 Internet 底层架构的研究。Lucene 早先发布在作者自己的 http://www.lucene.com/，后来发布在 SourceForge，2001 年年底成为 Apache 软件基金会 jakarta 的一个子项目：http://jakarta.apache.org/lucene/。

作为一个开放源代码项目，Lucene 从问世之后，引发了开放源代码社群的巨大反响。程序员们不仅使用它构建具体的全文检索应用，而且将之集成到各种系统软件中去，以及构建 Web 应用，甚至某些商业软件也采用了 Lucene 作为其内部全文检索子系统的核心。Apache 软件基金会的网站使用了 Lucene 作为全文检索的引擎，开源软件 Eclipse 的 2.1 版本中也采用了 Lucene 作为帮助子系统的全文索引引擎。Lucene 以其开放源代码的特性、优异的索引结构、良好的系统架构获得了越来越多的应用。

在 SaaS 应用中，与传统的互联网应用使用 Lucene 不同的是，每个租户的数据都应该是彼此隔离的，每个租户只需要也只应该搜索到自己的数据。基于数据隔离和系统架构伸缩性考虑，SaaS 应用的索引一般要按照租户来分别建立。图 4-4 展示了基于 Tenant 的索引构建和搜索的系统架构。

4.2.4 异步操作

异步操作并没有从本质上提升系统性能或者数据吞吐量。但是，异步操作使得用户体验得到了极大的提升，同时也能降低系统的并发负载。因此，将同步操作变成异步操作也是提升性能的一种重要的手段。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

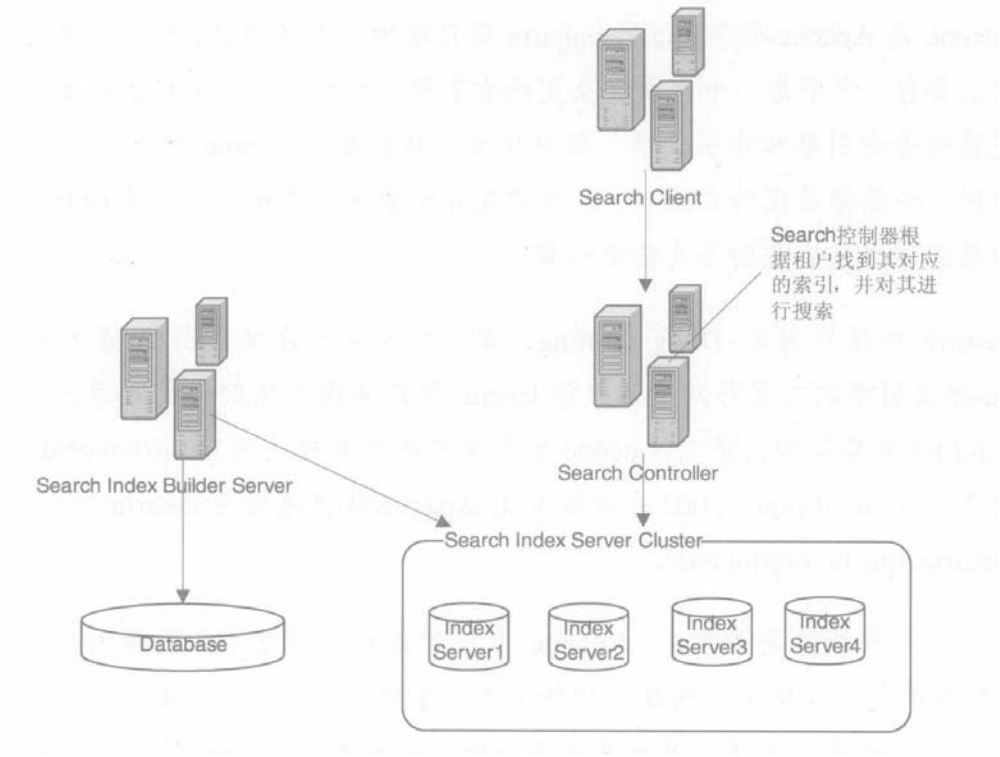


图 4-4　基于 Tenant 的索引构建和搜索的系统架构

异步操作有两个不同层面的异步：

- 表现层的异步，一般多采用 AJAX 技术来实现。
- 后台业务逻辑层面的异步，则更多的会采用 JMS、MQ 等技术。

对于部分耗时略长，但是用户又希望能够及时查看到结果的功能（如比，侠客录 CRM 中的发送邮件的功能），就适合采用 AJAX 技术来实现。这种方式一般适合用在表现层，用于提升用户体验，避免由于同步请求过多而引起的整个页面刷新。常见的 AJAX 框架有 DWR、Dojo 等。由于本书不是专门介绍 AJAX 技术的，所以在此不过多介绍，有兴趣的读者可以看《AJAX IN Action》等 AJAX 方面的书。

对于耗时可能很长，而用户和系统并不需要立即关注其处理结果的业务逻辑，则应该更多地使用后台业务逻辑层面的异步，即，异步消息机制。在 Java 领域，就是使用基于 JMS 规范的消息中间件。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在侠客录 CRM 中，就有两个场景分别用到了 AJAX 技术和 JMS 技术：

- 侠客录 CRM 中有一个给客户发送邮件的功能。由于发送邮件的功能一般耗时比较长，因此郭靖在展现层使用了 AJAX 技术，将邮件的发送变成一个 AJAX 请求。发送邮件时，页面不进行刷新，而仅仅浮出一个正在发送的层，以提升用户体验（如图 4-5 所示）。

- 侠客录 CRM 中有一个类型的订单会直接采用支付宝进行结算，因此生成该订单的时候，同时需要创建一笔支付宝的交易。这个过程中涉及调用支付宝的接口，一般适合采用 JMS 等异步机制进行，以免由于外部系统的原因而导致系统变慢甚至阻塞。

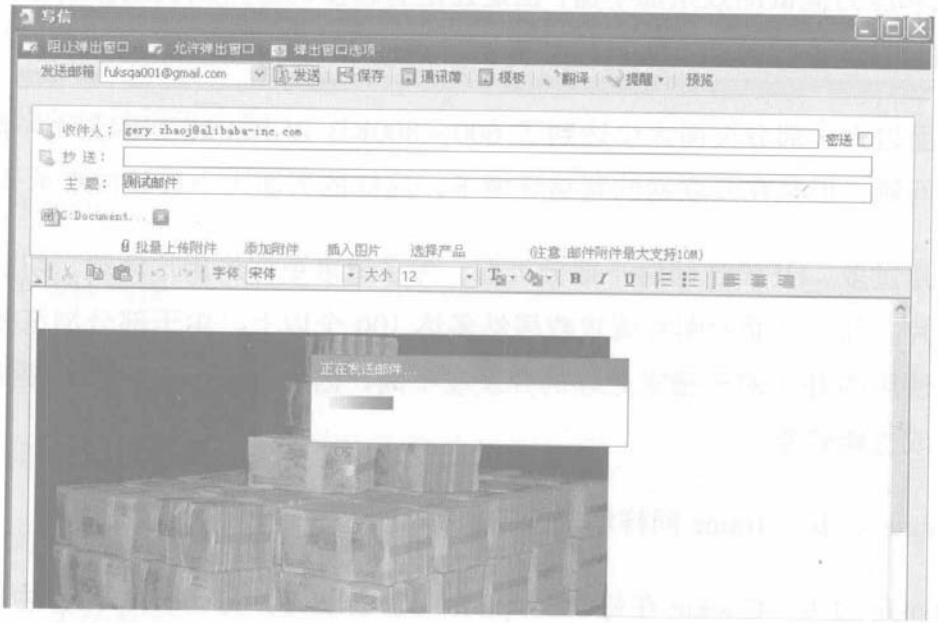


图 4-5　侠客录 CRM 中采用 AJAX 技术的邮件发送功能

4.3 Web 层性能优化

SaaS 应用作为互联网 Web 应用的一种，除了以上优化措施，还有很多适用于 Web 应用的通用性的一些优化方法。在这方面虽然做过多年软件开发，但显然还不谙于此道

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

的郭靖找到了一个做过多年网站的同学尹志平，尹志平同学很快就给了郭靖 Web 层性能优化方面的几点建议。

4.3.1 Web 开发的性能优化策略

之前的侠客录 CRM 虽然也是 B/S 架构的应用，但是由于其面向企业客户，大部分时候都部署在企业内网中，其在 Web 本身上并没有过多地关注其性能问题。在改造成 SaaS 版本之后，其实际上也成为了一种互联网的 Web 应用，众多的企业客户也都是通过 Internet 来使用它的。对于 Web 应用来说，不同地区的客户、不同的网络情况对于客户的访问速度都存在巨大的影响。这时候，虽然通过前面各种策略的服务端优化之后，且内部测试和压力测试的效果都不错，但是还是有较多的客户反映跟郭靖自己所测试看到的结果有较大的差异。尹志平很快分析了问题，并找出了几个重要的原因：

- 页面过大：部分页面大小达到了 600～800KB 以上。这在局域网中也许不算什么问题，但是在互联网的网络环境下，这样的页面大小显然很难承受。
- 图片过多：UI 设计师对于部分页面，为了追求更佳的用户体验，切割了大量的图片。部分页面的图片请求数居然多达 100 个以上，由于部分浏览器（例如广泛使用的 IE）对于请求处理的并发度不高，因此过多的请求会需要排队，从而造成性能较差。
- iframe 过多：iframe 同样增加了请求数。
- Cookie 过大：Cookie 在每个 Request 时都会上传，每个 Response 中都会下载。尤其是 Cookie 过大可能导致发起请求更慢，因为目前国内网络接入所广泛采用的 ADSL 技术在上传和下载速度上并不对称，上传速度会更慢。Cookie 过大与 iframe 过多两个问题叠加之后，问题会更加严重。

除了这些问题，尹志平还给了郭靖在 Web 开发性能优化方面的武功秘籍——《葵花宝典》，即著名的 Yahoo!团队实践分享——网站性能优化的 34 条黄金守则（详细的原文地址：http://developer.yahoo.com/performance/rules.html）。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间：可以联系群主获取更多大型企业内部技术教程。

1. 内容类

- 减少 Http 请求；
- 减少 DNS 查找；
- 避免跳转；
- 缓存 AJAX；
- 延迟加载；
- 提前加载；
- 减少 DOM 元素数量；
- 用域名划分页面内容；
- 减少 frame；
- 避免 404 错误。

2. 服务器类与 Cookie 类

- 使用 CDN；
- 为文件头制定 Expires 或 Cache-Control；
- GZip 压缩网页；
- 配置实体标签；
- 尽早刷新输出缓冲；
- 使用 Get 完成 AJAX 请求；
- 减小 Cookie 体积；

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 对于页面内容使用无 Cookie 域名。

3. JavaScript、CSS、图片类

- 把脚本置于页面底部；
- 使用外部 JavaScript 和 CSS；
- 削减 JavaScript 和 CSS；
- 剔除重复脚本；
- 减少 DOM 访问；
- 开发智能事件处理程序；
- 把样式表置于顶部；
- 避免使用 CSS 表达式（Expression）；
- 使用外部 JavaScript 和 CSS；
- 用<link>代替@import；
- 避免使用滤镜；
- 优化图像；
- 优化 CSS Sprites；
- 不要在 HTML 中缩放图像；
- favicon.ico 要小而且可缓存。

4. 移动应用类

- 保持单个内容小于 25KB；

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 打包组件成复合文本。

虽然这些优化法则更多的是侧重于互联网网站的优化，不过很多内容对于基于 Web 的 SaaS 应用同样具有参考价值。

4.3.2 Http 服务器的性能优化策略

侠客录 CRM 原有架构中直接使用 JBoss 作为应用服务器和 Http 服务器。JBoss 作为专业的应用服务器，虽然也可以提供 Http 服务，但却并不是一个专业的 Http 服务器。JBoss 在性能、安全性等方面，与专业的 Http 服务器相比还是具有较多的缺陷的。

Http 服务器首选当然是大名鼎鼎的 Apache。

Apache 是世界排名第一的 Web 服务器，根据 netcraft（www.netsraft.co.uk）所作的调查，世界上 50%以上的 Web 服务器在使用 Apache。

1995 年 4 月，最早的 Apache（0.6.2 版）由 Apache group 公布发行。Apache group 是一个完全通过 Internet 进行运作的非盈利机构，由它来决定 Apache Web 服务器的标准发行版中应该包含哪些内容。

Apache 的特性：

（1）几乎可以运行在所有的计算机平台上；

（2）支持最新的 HTTP1.1 协议；

（3）简单而且强有力的基于文件的配置（httpd.conf）；

（4）支持通用网关接口（CGL）；

（5）支持虚拟主机；

（6）支持 Http 认证；

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（7）集成 Perl；

（8）集成的代理服务器；

（9）可以通过 Web 浏览器监视服务器的状态，可以自定义日志；

（10）支持服务器端包含命令（SSL）；

（11）支持安全 Socket 层（SSL）；

（12）具有用户会话过程的跟踪能力；

（13）支持 fastcgi；

（14）支持 Java Servlets。

Apache 主要有 Prefork、Worker、Event 三种工作模式。不同工作模式具有一些不同的特性，选择哪种模式对服务器的性能有较大的影响。表 4-3 对这三种工作模式进行了一些对比。

表 4-3　Apache 的三种工作模式比较

工作模式	特　性
Prefork	■ 多进程服务方式 ■ 单独进程处理连接请求 ■ 一个连接对应一个工作进程 ■ 性能差
Worker	■ 多进程多线程服务方式，或者单进程多线程服务方式 ■ 每个工作进程配置固定数量的工作线程 ■ 没有专门的线程处理网络事件 ■ 一个连接对应一个工作线程
Event	■ 基于 Worker 模式的改进 ■ 有一个专门的线程处理网络事件 ■ 工作线程使用阻塞模式处理读请求

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

除了工作模式的选择之外，还有如下的优化策略可供选择：

- 使用最新稳定版本的 Apache，新版本通常具有更高的性能提升和更好的安全性。例如，上面的三种工作模式中的 Event 模式就只有在版本 2.2.X 中才存在。
- 在 httpd.conf 中设置“HostNameLookups off”能避免针对每个访问者的 DNS 域名的反向查询。
- 保持你的 Apache 苗条，编译那些仅仅需要的模块。
- 采用独立的服务器处理图片请求。

除了 Apache 之外，近年来 Lighttpd 也是被广泛采用的 Http 服务器，其在处理静态资源请求方面相比 Apache 具有更好的性能。但是由于其对于动态内容的支持并不佳，对于大部分的 SaaS 应用，以动态内容居多，因此会较少使用。

4.4 性能监控

发现问题之后，找出对应的解决方案并不难。郭靖对于自己处理问题的能力相当自信，但是要及时地发现这些问题并不容易。之前的问题大多是客户反馈之后再去进一步跟进，而这时该问题给客户造成的负面影响已经大规模爆发了。因为，大部分客户都是在忍无可忍的情况下才会来反映这些问题的。

在这一次的性能优化过程中，郭靖也采取了一些措施来主动发现性能问题。比如，通过 MySQL 的 slow query 进行耗时 SQL 统计，对于一些关键场景进行模拟压力测试等。应该说这些措施都非常有效，但却不是足够的。例如，用户客户端相关的性能问题，典型的包括网络流量、客户端 JS 等引起的性能问题，就很难发现；再如，模拟的压力测试场景可能与实际生产环境存在较大的差异等。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在这种情况下，如果能有一个工具能够实时地统计出应用在各种场景下（不同地区、不同用户、不同页面）的性能数据并及时预警该有多好。郭靖的想法并非幻想，其实有人已经实现，有图为证（见图 4-6 和图 4-7）。



图 4-6　每日最慢前 10 数据查询



图 4-7　每日统计数据查询

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

图 4-6 和图 4-7 是阿里软件性能中心的部分页面的截图，该性能中心可以统计出每个请求的服务端响应时间、网络时间和客户端加载时间，并且可以按照页面、用户所在地区、时间等多个维度进行统计分析。截至本书写作之时，该工具还是阿里软件的 SaaS 软件内部使用的工具。不过相信在读者朋友看到这本书的时候，该性能中心已经面向众多的 SaaS 软件开发商开放了，帮助大家监控自己的应用性能（大家如果想知道更多的阿里软件的 SaaS 平台的内容，可以详细阅读本书最后一章“开放的 SaaS 平台”）。

4.5 优化的效果

经过一系列的性能优化，郭靖看到应用服务器和数据库服务器的压力已经显著下降。即使在已经达到 1000 个用户的情况下，应用服务器和数据库服务器的负载都稳定在 20%左右，对支撑杨康要求的 2000 个用户的目标应该是没有什么大问题了。

杨康：最近我们的系统反应挺快的嘛，比前段时间强多了。

郭靖：那是，我们可是花了好几个通宵，想了各种各样的方法来解决我们的性能问题，不要说 2000 个用户，就算是年底达到 3000 个用户，我们也没有问题。

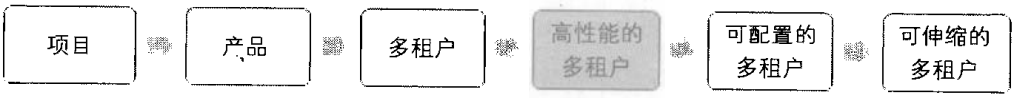
杨康：那好，强的！那接下来就看我的吧，确保 3 个月 2000 个用户，年底到 3000 个用户。

杨康与郭靖击掌欢庆之后，杨康又开始奔赴下一个战场，而郭靖则又回到电脑前面考虑下一步的改进计划。

4.6 小结

又前进了一步，我们的 SaaS 征程又完成了一个至关重要的里程碑：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



这一章中我们简单介绍了多租户架构下的性能优化方面的最佳实践：

- 数据库层的性能优化
 - 建立合适的索引；
 - 消除大数据量表连接；
 - 避免复杂 SQL。
- 应用层的性能优化
 - Cache；
 - 报表计算；
 - 基于 Tenant 的索引搜索；
 - 异步操作。
- Web 层的性能优化
 - Web 开发的性能优化策略；
 - Http 服务器的性能优化策略。

当然，由于本书并非专门的介绍数据库优化、Cache 或者搜索引擎的书籍，因此很多内容都是浅尝辄止，更多的可能是给读者朋友一个前进的方向。如果有读者对于具体内容有更深的应用，则可能需要查阅更多的书籍和资料以进一步了解。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

此外，如果读者朋友接触过互联网应用的开发，就会发现本章的许多最佳实践其实都是互联网应用开发通用的最佳实践，并不局限于 SaaS 应用。确实如此，SaaS 应用作为一种多租户的应用架构，其更多类似于互联网应用，或者说就是互联网应用的一种。因此，大多数互联网应用的最佳实践，在 SaaS 应用中也都能够适用。对于传统软件的开发者，从单租户到多租户的架构转换过程中，思维方式的转变其实是最困难的。这就是互联网时代给软件业带来的革命，我们软件业的同行们需要在这种革命中转变思维，适应变化，甚至引导这场革命的发展方向。

SaaS 架构设计

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第5章 Multi-Tenant 应用的可配置性

5.1 数据可配置……100
5.1.1 定制字段……101
5.1.2 预分配字段……102
5.1.3 名称值对……104
5.1.4 方案比较……106
5.2 功能可配置……107
5.2.1 原子功能划分……108
5.2.2 功能包设计……111
5.2.3 功能使用校验……115
5.3 界面可配置……116
5.3.1 系统菜单可配置……116
5.3.2 页面元素可配置……118
5.4 流程可配置……119
5.5 配置元数据管理……119
5.5.1 配置元数据……120
5.5.2 租户配置数据……123
5.5.3 配置元数据服务……125
5.6 可配置系统运行……125
5.6.1 系统菜单框架……126
5.6.2 功能页面容器……127
5.6.3 扩展数据引擎……128
5.6.4 功能引擎……128
5.7 小结……128

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

一日，杨康兴冲冲地直奔郭靖办公室。

杨康：呵呵，我们又有新客户了！

郭靖：那太好了。

杨康：不过客户有很多新想法哦。

郭靖：什么想法，说来听听。

杨康：他们想保存客户的一些特别信息，例如，工资级别、工作种类……

郭靖：这样啊，那又得改我们的系统了，我们的系统已经针对好多家客户做这种定制需求了，上次就加了客户来源和客户介绍人两个字段，数据表已经改得乱七八糟的，逻辑判断也弄得很复杂了。

杨康：是啊，经常有客户给我们提这种需求，我们要能支持可以配置的就好了，客户想要保存什么数据，就配什么数据，那多好啊！

郭靖：支持可配置？这个主意不错，只是支持数据可配置还不够，界面也得可以配置，才有地方可以输入数据和展示数据。

杨康：还有，我们系统的功能这么多，可很多客户都不需要这么多功能，也不想花这么多钱。虽然我们的系统已经很便宜了，但花钱买了，又用不着，客户心里总是不爽的。

郭靖：这得实现功能可以配置。

郭靖一边说着，一边陷入了深思：如何在系统内实现客户的可配置性呢？

在传统的应用系统中，也会考虑到可配置的问题。这主要是为了解决在设计时一些不能完全确定的参数，应对需求的细微变更，适应运行环境的信息配置等。这些配置在 SaaS 应用中，仍然是存在的。这种配置是针对整个系统的，在系统设计和实现上，仍可以参照传统系统的实现方案，在此不做详细讨论。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

本章要讨论的可配置性是 SaaS 应用针对多租户的各种可配置性的实现。根据上面杨康与郭靖的对话，可以分析出针对租户的可配置性，大致包括以下几个方面的内容：

- 数据可配置；
- 功能可配置；
- 界面可配置；
- 流程可配置。

这种针对租户的配置与传统的针对产品的配置相比，在许多方面都存在很大区别。详细信息可参见表 5-1。

表 5-1　产品配置与租户配置比较

比 较 项	产 品 配 置	租 户 配 置
相关度	与产品特征相关	与租户特征相关
配置时间	一般在系统运行前配置	在系统运行过程中配置
数量	系统内只有一份	一个租户一份配置
使用人员	由运行、维护或实施人员配置	由租户管理员来修改和配置
作用范围	整个系统内有效	对租户数据有效
装载时机	系统初始时装载	使用时动态装载

以上这些差别，使得系统在架构和设计方案上变得更加复杂。本章将重点对这些方案进行逐一分析介绍。

5.1 数据可配置

正如上面杨康所说的，有租户希望能够保存客户的纪念日、客户来源等，租户总是会有一些在其他租户看来是多余的数据需要保存起来，而又往往很难说服客户不要这种需求。这种系统实现过滤中并不存在，而用户又需要保存的数据，称之为扩展数据。显

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

然，不同的客户希望能保存的扩展数据可能是完全不同的。

对于扩展数据的处理，在传统的应用中，由于针对不同客户部署了不同的应用实例，这样就可以定制各自的实例，增加单个客户的扩展数据，来满足其个性化需求。在多租户的 SaaS 应用中，所有的租户都在使用同一个应用实例，在同一个应用实例上又如何满足大量租户的各自不同的扩展数据需求呢?

最常见的解决办法就是实现扩展数据的可配置，这正是本节要重点讨论的数据可配置。实现数据可配置有以下 3 种常见方案:

- 定制字段;
- 预分配字段;
- 名称值对。

下面就对这 3 种方案分别进行介绍，并对各自的优缺点进行比较。

5.1.1 定制字段

定制字段是传统应用采用的最常见的解决方案，就是根据客户的需求在数据表上增加相应的定制字段来保存扩展数据。例如，根据租户 33 的需求，在客户表中增加来源 Source 和介绍人 Introducer 两个字段，详细信息可参见表 5-2。

表 5-2 定制字段数据示例

Id	TenantId	Name	Age	…	Source	Introducer	salary	work
111	33	Tom	25		google	John		
112	33	Jack	22		google	Tom		
113	33	Mary	40		yahoo	Jack		
114	40	Joy	30				3000	Teacher
115	40	Tony	32				5000	Engineer
…	…	…	…					

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

从实现的角度来看，这是一种最简单的方案，不需要处理复杂数据延伸的跟踪。对于传统应用来说，这种方案也是可以接受的。因为每个客户都有自己独立的数据库，增加的字段不会影响到其他的客户。但对于多租户的 SaaS 应用来说，如果允许为每个租户都增加自定义的字段，这些字段将会多如牛毛。同时，这些字段对于其他租户来说，是没有任何意义的，而且严重破坏了数据表的结构。另外，这种方案需要针对每一个要定制数据列的租户，进行数据表字段的特殊处理。显然，定制字段并不是解决多租户 SaaS 应用下数据可配置的理想方案。

5.1.2 预分配字段

预分配字段是指在用户可能有扩展需求的表中预设一定数量的字段。当用户提出扩展需求时，使用其中的一个或多个字段来保存扩展数据。例如，租户可能需要保存客户的扩展数据，那么可以在客户表增加几个预分配字段。详细信息可参见表 5-3。

表 5-3 预分配字段数据示例一

Id	TenantId	Name	Age	…	ExtInt	ExtStr	ExtDate
111	33	Tom	25			Google	
112	40	Joy	30		3000	Teacher	
113	33	Jack	22			Google	
114	40	Tony	32		5000	Engineer	
115	33	Mary	40			yahoo	
116	56	John	35				2008-8-1

与前面的定制字段方案相比，都是在数据表增加字段来存取扩展数据，但这些预分配的字段是没有特定含义的。对不同的租户来说，这些字段保存的是不同含义的数据。例如，ExtStr 这个字段对于租户 33 来说，保存的是来源；但对于租户 40 来说，保存的却是职业。当然，对于同一租户来说，同一个扩展字段只能用来保存同一含义的数据。这些字段的数量也是有限的，不会随着租户数量的上升而不断增加。不过，具体预设多少个预分配字段，要根据具体的应用场景和扩展数据需求来设计。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 5-3 中总共预设了 3 个预分配字段，分别是一个整型字段、一个字符串字段和一个日期字段，这些字段和数据类型是强相关的。虽然设了 3 个预分配字段，但却保存不了租户 33 的来源 Source 和介绍人 Introducer 两个扩展字段，因为这两个字段都要求按字符串保存了，而上面只预分配了一个字符串字段。如果为每一种数据类型都设置足够多的预分配字段，又会造成许多无意义的空闲字段。

针对上面方案中存在的问题，为了更充分地利用预分配字段，可以将所有的扩展数据转换成字符串以后再来进行存储，这样预分配字段的数据类型就可以全部定义成字符串了。在预分配字段的总个数不大于单个租户要扩展的字段个数的情况下，就可以满足所有客户的数据可配置需求。详细信息可参见表 5-4。

表 5-4　预分配字段数据示例二

Id	TenantId	Name	Age	…	Ext1	Ext2	Ext3
111	33	Tom	25		Google		
112	40	Joy	30		3000	Teacher	
113	33	Jack	22		Google	Tom	
114	40	Tony	32		5000	Engineer	
115	33	Mary	40		yahoo	Jack	
116	56	John	35		2008-8-1		

从数据存储来看是没有问题的，但是展示数据的时候如何将数据转换成其原有的真实数据类型呢？在保存之前又如何进行数据类型的校验呢？同样都保存在 Ext1 字段，对于租户 33 来说，要转换成字符串类型；对于租户 40 来说，要转换成整数类型；而对于租户 56 来说，要转换成日期类型数据。对于租户使用各字段来保存的真实数据类型，可以由租户来配置，并作为配置元数据进行管理。在使用数据时，系统可以根据元数据的配置信息，再转换成真实的数据类型。

与数据类型一样，数据的含义对于不同的租户来说也是不一样的，同一个预分配字段所保存的是不同含义的数据。例如，同样都是保存在 Ext1 字段中的数据，对于租户 33 来说，保存的是来源；对于租户 40 来说，保存的是工资；而对于租户 56 来说，保存的却是纪念日。那么，对于各个租户使用各个字段的含义也是需要作为配置元数据进行

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

管理的。

对于预分配字段的含义和类型，就需要单独定义一个配置元数据的数据表来保存。对于不同的租户使用各表中的每一个扩展字段，分别保存一条配置记录。详细信息可参见表 5-5。

表 5-5　预分配字段元数据

ConfigId	TenantId	Table	Column	Content	DataType
20	33	custom	Ext1	Source	string
21	33	custom	Ext2	Introducer	string
22	40	custom	Ext1	salary	int
23	40	custom	Ext2	work	string
24	56	custom	Ext1	Commemoration Day	date
25	…				

有了表 5-5 保存的元数据，在使用租户的扩展数据时，就可以根据当前的租户 TenantId 和所在数据表 Table 查询到该租户使用了哪几个预分配字段，以及各字段的数据类型和数据含义。这样，就可以将扩展数据转换成相应的数据类型来使用，并在界面上展示该扩展数据的含义，真正实现数据的可配置。

5.1.3　名称值对

前面介绍的预分配字段方案，在一定程度上很好地实现了数据的可配置性。但预分配字段的个数在系统设计时就必须确定，这对于 SaaS 应用会面临形形色色的租户来说，还是一个很困难的问题。如果预分配字段设计得太多，就会造成数据库表膨胀，还会产生很多无意义的空闲字段，造成浪费，从而引发数据库性能下降等问题。如果预分配得太少的话，更多用户的扩展数据就没有地方可以保存，无法满足扩展数据可配置的需求。为了解决这个问题，可以考虑将扩展数据的保存与原数据表分离，另外用一张统一的扩展数据表来保存。

扩展数据表将数据表的横向扩展列转换成纵向的数据集，将每一条原数据记录的每

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

一个扩展字段，都保存成一条扩展数据行。将数据表中的数据记录与配置元数据表中的配置记录关联，构成扩展数据记录。具体关联结构可参见图 5-1。

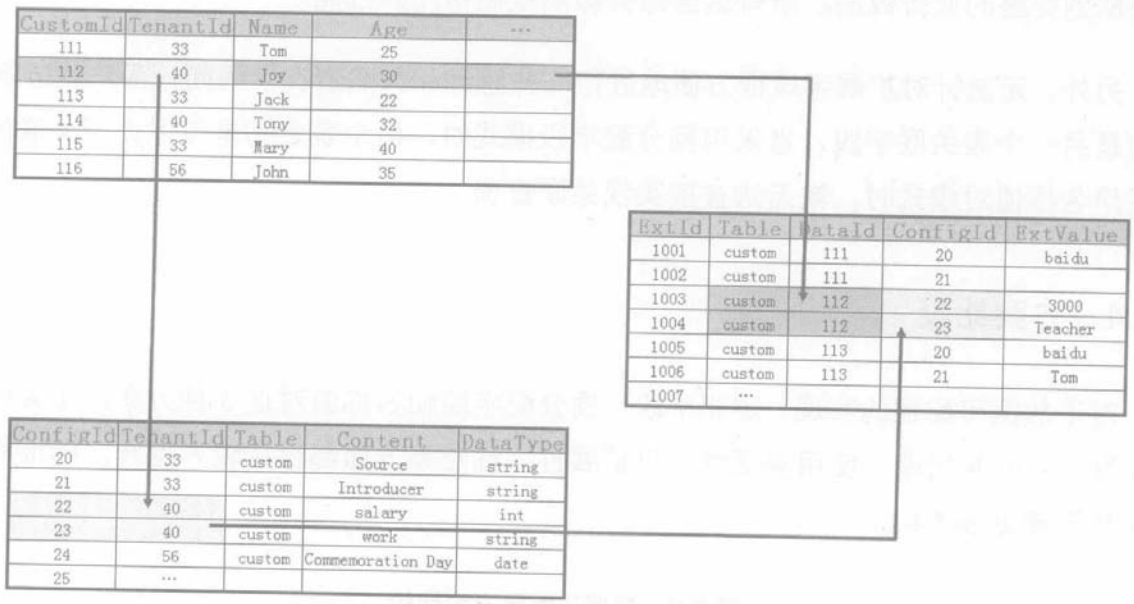


图 5-1　名称值对关联结构图

在配置元数据表中所配置的扩展数据与业务数据表中的租户 TenantId 是相对应的；扩展数据表的 DataId 与业务数据表中的主键是相对应的；而扩展表的 ConfigId 与配置元数据表中配置扩展项是相对应的。与租户配置的扩展数据项个数相对应，该租户的每一条业务数据都会有相应个数的扩展数据被保存。例如，租户 40 针对 custom 数据定义了两个扩展数据 22（salary）和 23（work），那么与租户 40 的业务数据 112 对应就会有两条扩展数据 1003（salary：3000）和 1004（work：Teacher）。

名称值对这种模式，给扩展数据的保存提供了非常大的灵活性，可以提供无限数量的自定义扩展字段。租户可以根据业务需要不断地增加自己的自定义数据，每增加一种扩展数据，会在配置元数据表中增加一条配置记录。同时，与该租户的每一条业务数据相对应，会增加一条扩展数据到扩展数据表中。针对不同的租户，定义不同的扩展数据。在扩展表中只会保存该租户的扩展数据，不会再有空闲的字段，其他租户也不会因此占有多余的空间。可见，这种模式并不会造成资源的大量浪费。

SaaS 架构设计
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

当然，这种模式在带来灵活性的同时，也增加数据操作的复杂性。新增、修改和删除数据时，都需要对自定义数据进行分拆处理。查询数据时，也需要多次访问数据库才能获取到完整的业务数据。所有这些都会影响数据访问的性能。

另外，无法针对扩展字段很方便地进行一些统计和关联查询。例如，客户的自定义数据是另一个表关联字段，当采用预分配字段模式时，两个表之间是可以进行关联的；但采用名称值对模式时，就无法直接实现关联查询。

5.1.4 方案比较

对于数据可配置的实现，定制字段、预分配字段和名称值对这 3 种方案，在实现的复杂度、空间利用率、使用灵活性、可扩展性、性能等方面都存在较大差异。详细的比较信息可参见表 5-6。

表 5-6 数据可配置方案比较

比 较 项	定 制 字 段	预分配字段	名 称 值 对
可扩展性	低	中	高
性能	高	中	低
灵活性	低	中	高
实现复杂度	低	中	高
空间利用率	低	中	高

在具体 SaaS 应用开发过程中，要根据实际应用对数据的扩展定制需求、租户的数量、开发能力及周期等，选择相应的方案。

本节主要讨论的是可配置数据的数据存储方案。对于可配置数据在系统中的展示与提交，具体实现方案将在后面的可配置系统运行一节中详细探讨。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

5.2 功能可配置

对于传统应用来说，应用的功能与客户的需求基本是能够相符的。如果要将同一个产品提供给需求差别比较大的多个客户使用，只要在部署的时候将客户不需要的功能去掉就可以了。但对于 SaaS 应用来说，需要面对为数众多的租户，系统的功能要与每一个租户的需求相符是一件基本上不可能的事情。大部分的租户可能都只会使用到应用中的部分功能。而所有的租户使用的是同一个应用，不可能像传统的应用一样，为每一个客户部署一个不同版本的应用。SaaS 应用如何允许不同的租户购买不同的功能集合，却同时在线使用呢?

这也正是 SaaS 应用强调的“按需使用，按需付费”。也就是说，多租户 SaaS 应用要能够支持让租户有选择地购买自己需要的功能，不同的租户可以同时使用不同的功能集合。这样，就可以很好地解决不同的租户需求不完全重合的问题，更大程度地适应来自不同行业的租户之间功能需求的差异。例如，对于 CRM 系统来说，教育、培训、服务、饮食等行业就并不是很重视产品管理，而且很少有订单管理的概念。相对来说，他们更重视客户服务、行程跟踪的管理。相反，零售、分销、制造等行业会重视产品管理和订单管理，却很少强调客户服务及行程管理等。

另外，SaaS 应用提倡的“按需使用，按需付费”的另一个含义是，同一个租户在不同时期，可能会有不同的需要，可以选择不同的功能集合，并为之付费。例如，租户一开始可能只需要使用系统中最基本的功能和免费的功能，随着业务扩展的需要和对系统的熟悉，可以选择购买更多的功能。反之亦然，租户一开始可能购买了很多的功能，使用以后才发现许多功能用不上，那么，在其续费的时候，可以去掉那些用不上的功能，选择较少的功能集合。

因此，SaaS 应用需要支持功能可配置，允许租户有选择地购买系统中的功能。租户需要哪些功能就购买哪些功能，而租户没有购买的功能是使用不了的。通过在系统运行时的在线配置，使得不同租户可以同时在线使用系统的不同功能。

SaaS 应用要如何才能做到功能可配置呢?

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

5.2.1 原子功能划分

要实现功能可配置，首先需要将整个系统的功能进行分解。整个应用需要分解成一个个最基本的、相对独立、互不重叠的原子功能。当所有的原子功能叠加起来，就是整个应用所提供的全部功能。

1. 功能分解

功能分解就是将整个系统的功能分解成最基本的相对独立的原子功能。这也是进行功能配置的基础。进行功能分解时，应遵循以下几个原则：

- 每个功能都是有价值的；
- 每个功能都不可再细分；
- 功能间不相互重叠；
- 功能之间不循环依赖；
- 整个系统功能是完整的。

功能分解要遵循用户价值驱动的原则。也就是说，每一个原子功能都要能够给用户带来某方面的价值。如果原子功能是不能给用户带来价值的，就不会有用户愿意花钱购买。显然，这样一个功能就是没有价值和意义的。例如，在 CRM 系统中，新建客户或修改客户时，会包含客户账号校验、客户联系方式维护等，这些只是新建客户功能或修改客户功能的一个步骤。这个步骤并不能给用户带来任何价值，也就不能单独定义成一个原子功能。

在保证原子功能对于用户有价值的前提下，功能分解要尽量细化，直到不能再细分为止。例如，在 CRM 系统中，用户在创建订单过程中，需要依次新建订单和提交订单，这两个操作对于用户来说都是有意义的。那么仅定义一个创建订单的原子功能就是合适的，将其拆分成新建订单和提交订单两个原子功能，会更清晰、更灵活。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在进行功能分解时，要关注原子功能之间的关系。功能之间不要相互重叠，也不要相互依赖。

功能分解要保证整个系统的功能完整性。也就是说，功能分解划分出来的所有原子功能，要能覆盖整个系统的所有功能，而不存在没有被划分的系统功能。尤其是对于一些比较独立的功能更需要关注。例如，在 CRM 系统中，客户数据的导入和导出功能，在功能分解时就很容易被忽略。

根据以上原则，对 CRM 系统进行功能分解，详细信息可参见图 5-2。

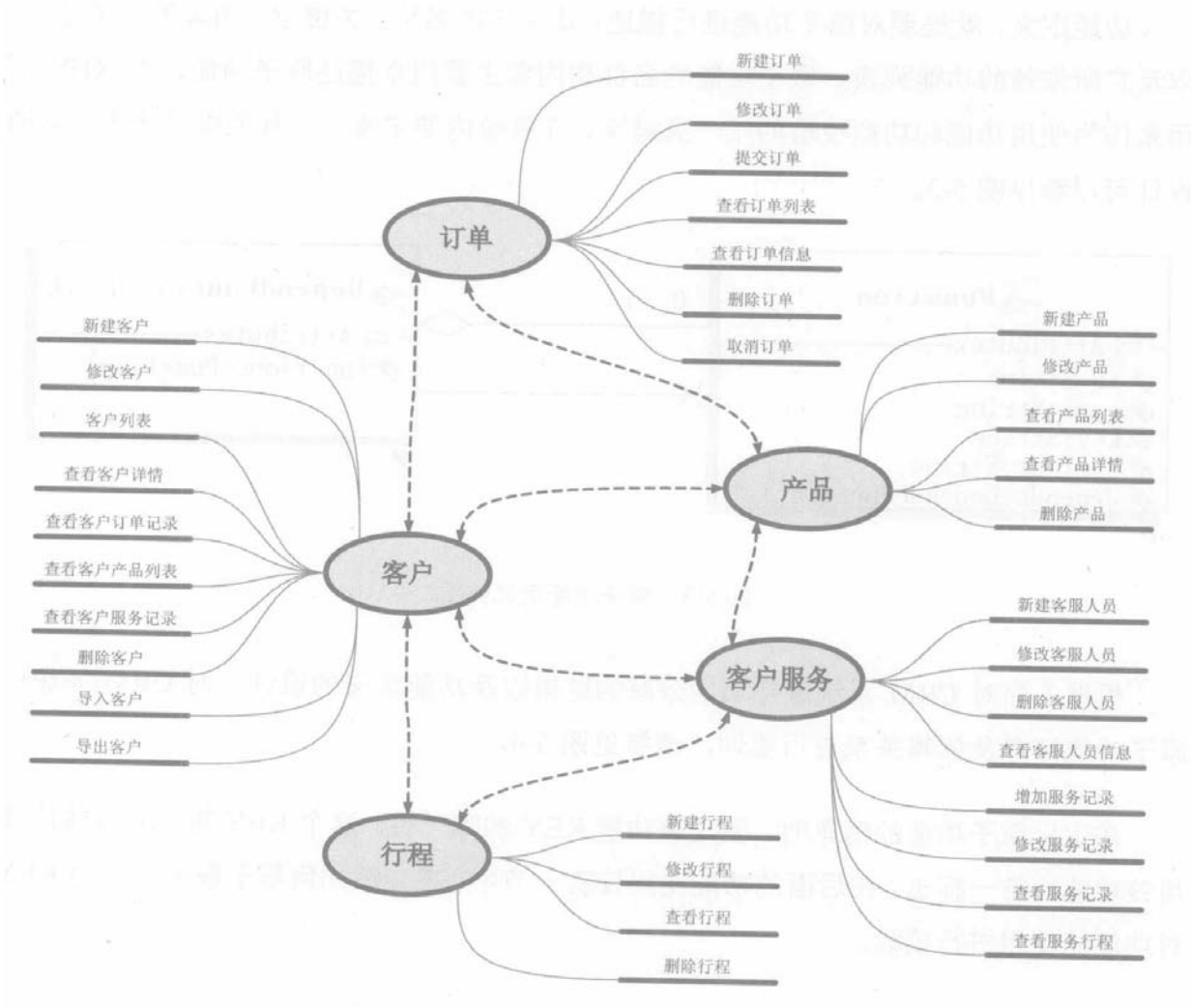


图 5-2　CRM 系统原子功能分解图

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

2. 功能定义及依赖

从上面功能分解的结果可以发现，并不是所有的原子功能都可以单独使用。有些功能是需要依赖其他功能才能使用的，功能之间是存在一定的依赖关系的。例如，在 CRM 系统中，“查看客户产品列表”功能依赖于“查看产品列表”功能，如果租户没有购买“查看产品列表”这个功能，那么“查看客户产品列表”也是不能使用的。所谓功能依赖，就是指一个功能在没有另外某个或某些功能的情况下是不能使用的。而一个功能可能依赖另外的一组功能。

功能定义，就是要对原子功能进行描述，定义它的名称、关键字、内容等相关信息，以及它所依赖的功能列表。原子功能的名称和内容主要用来描述原子功能，而 KEY 则用来作为使用功能和功能校验的唯一关键字，在系统内要求唯一。有关原子功能定义的设计可以参见图 5-3。

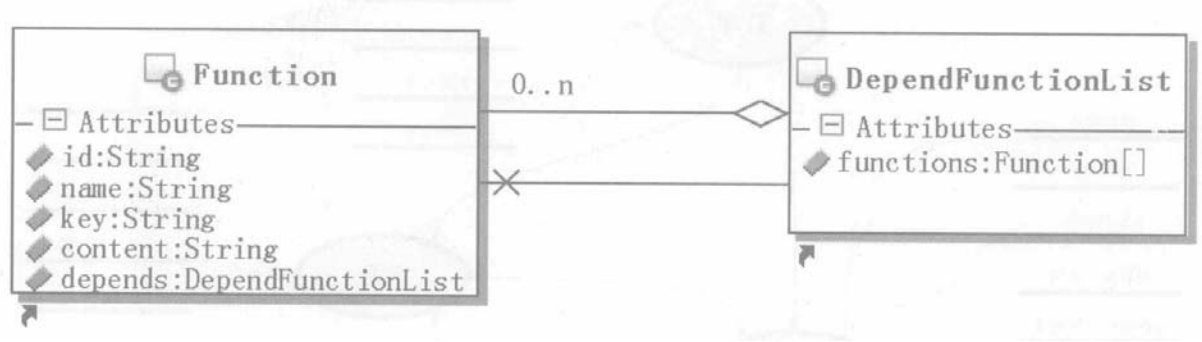
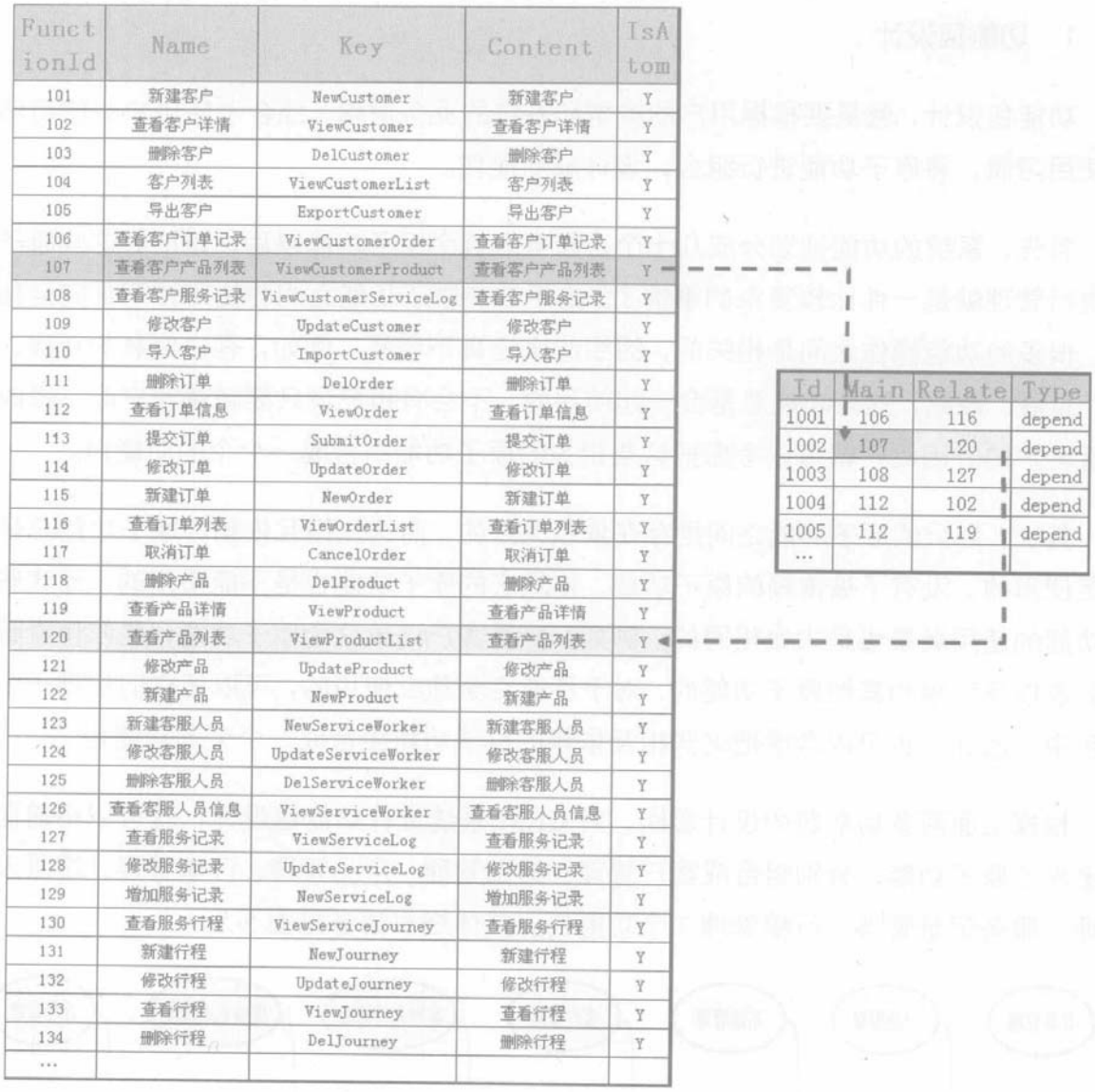


图 5-3　原子功能定义类图

根据上面对 CRM 系统进行功能分解的结果以及功能定义的设计，对 CRM 系统的原子功能信息及依赖关系进行整理，请参见图 5-4。

在定义原子功能的信息时，要注意功能 KEY 的唯一性，这个 KEY 将作为系统内使用该功能的唯一标志。在后面的功能使用校验一节中，将介绍如何基于原子功能的 KEY 对功能的使用进行校验。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



FunctionId	Name	Key	Content	IsAtom
101	新建客户	NewCustomer	新建客户	Y
102	查看客户详情	ViewCustomer	查看客户详情	Y
103	删除客户	DelCustomer	删除客户	Y
104	客户列表	ViewCustomerList	客户列表	Y
105	导出客户	ExportCustomer	导出客户	Y
106	查看客户订单记录	ViewCustomerOrder	查看客户订单记录	Y
107	查看客户产品列表	ViewCustomerProduct	查看客户产品列表	Y
108	查看客户服务记录	ViewCustomerServiceLog	查看客户服务记录	Y
109	修改客户	UpdateCustomer	修改客户	Y
110	导入客户	ImportCustomer	导入客户	Y
111	删除订单	DelOrder	删除订单	Y
112	查看订单信息	ViewOrder	查看订单信息	Y
113	提交订单	SubmitOrder	提交订单	Y
114	修改订单	UpdateOrder	修改订单	Y
115	新建订单	NewOrder	新建订单	Y
116	查看订单列表	ViewOrderList	查看订单列表	Y
117	取消订单	CancelOrder	取消订单	Y
118	删除产品	DelProduct	删除产品	Y
119	查看产品详情	ViewProduct	查看产品详情	Y
120	查看产品列表	ViewProductList	查看产品列表	Y
121	修改产品	UpdateProduct	修改产品	Y
122	新建产品	NewProduct	新建产品	Y
123	新建客服人员	NewServiceWorker	新建客服人员	Y
124	修改客服人员	UpdateServiceWorker	修改客服人员	Y
125	删除客服人员	DelServiceWorker	删除客服人员	Y
126	查看客服人员信息	ViewServiceWorker	查看客服人员信息	Y
127	查看服务记录	ViewServiceLog	查看服务记录	Y
128	修改服务记录	UpdateServiceLog	修改服务记录	Y
129	增加服务记录	NewServiceLog	增加服务记录	Y
130	查看服务行程	ViewServiceJourney	查看服务行程	Y
131	新建行程	NewJourney	新建行程	Y
132	修改行程	UpdateJourney	修改行程	Y
133	查看行程	ViewJourney	查看行程	Y
134	删除行程	DelJourney	删除行程	Y
...				

Id	Main	Relate	Type
1001	106	116	depend
1002	107	120	depend
1003	108	127	depend
1004	112	102	depend
1005	112	119	depend
...			

图 5-4　原子功能数据示意图

5.2.2　功能包设计

当系统的功能被划分成许多原子功能以后，就可以让用户想使用哪个功能就使用哪个功能了吗？事实上并不是这么简单。针对每一个租户的需求，来配置其购买和使用的原子功能，显然是一件很复杂的事情，也是很不现实的。比较好的做法，就是根据用户的类型和使用的场景，对原子功能进行打包，然后为每一个用户挑选其合适的功能包。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

1. 功能包设计

功能包设计，就是要根据用户的类型和系统的业务逻辑，综合考虑用户的使用场景和使用习惯，将原子功能进行组合，设计成功能包。

首先，系统的功能被划分成几十个，甚至上百个原子功能以后，直接对这些原子功能进行管理就是一件比较复杂的事情了。由于这些原子功能之间的使用并不是完全独立的，很多的功能操作之间是相关的，组织起来还真不容易。例如，客户信息的新建、修改、查看、删除，这些功能总是会一起使用的，不会有租户说只要新建客户而不修改和查看客户的。因此，就可以考虑把这些相关的原子功能组合成一个个的功能包。

其次，划分的原子功能之间是存在依赖关系的，而这些相互依赖的原子功能总是要绑定使用的。没有了被依赖的原子功能，依赖它的原子功能也是不能使用的；而这些原子功能的使用场景也是大致相同的。例如，查看客户订单这个原子功能，是依赖查看订单列表以及订单的其他原子功能的，对于不能完全独立使用的，可以把它们放到一个功能包中。因此，也可以考虑把这些相互依赖的原子功能组合成一个个的功能包。

根据上面两条功能包的设计思路，对 CRM 系统进行功能包设计，就可以将前面划分出来的原子功能，分别组合成客户管理、基础管理、产品管理、订单管理、客服人员管理、服务记录管理、行程管理 7 个功能包。具体细节请参见图 5-5。

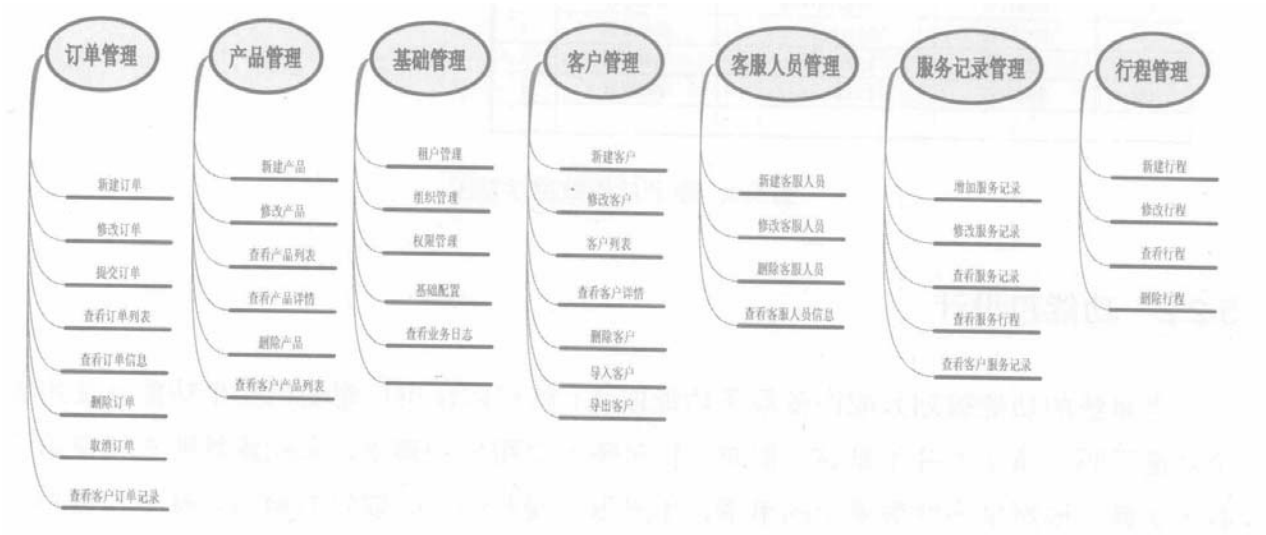


图 5-5　CRM 系统功能包设计示意图

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

功能包的设计要遵循高内聚、低耦合的原则，尽量将相关的和相互依赖的原子功能设计在一个功能包中。同时，应该尽量减少功能包与功能包之间的依赖，使得各个功能包尽可能独立地进行操作使用。

2. 销售包设计

通过功能包的设计，已经将系统功能组合成几个相对比较独立的部分了，但这些功能包仍然不可以完全独立使用。为了让客户购买以后能够充分地使用系统，还需要按不同的商业意图构造适宜于用户使用的销售包。

销售包只是一种以向用户进行销售而定义的一种功能包。例如，上面的 CRM 系统，按用户需要使用功能的多少，可以划分为最小版、标准版、完整版；按客户所属的行业类型，可以划分为服务行业版、制造行业版。各个销售包所包括的功能包及原子功能的关系，请参见图 5-6。

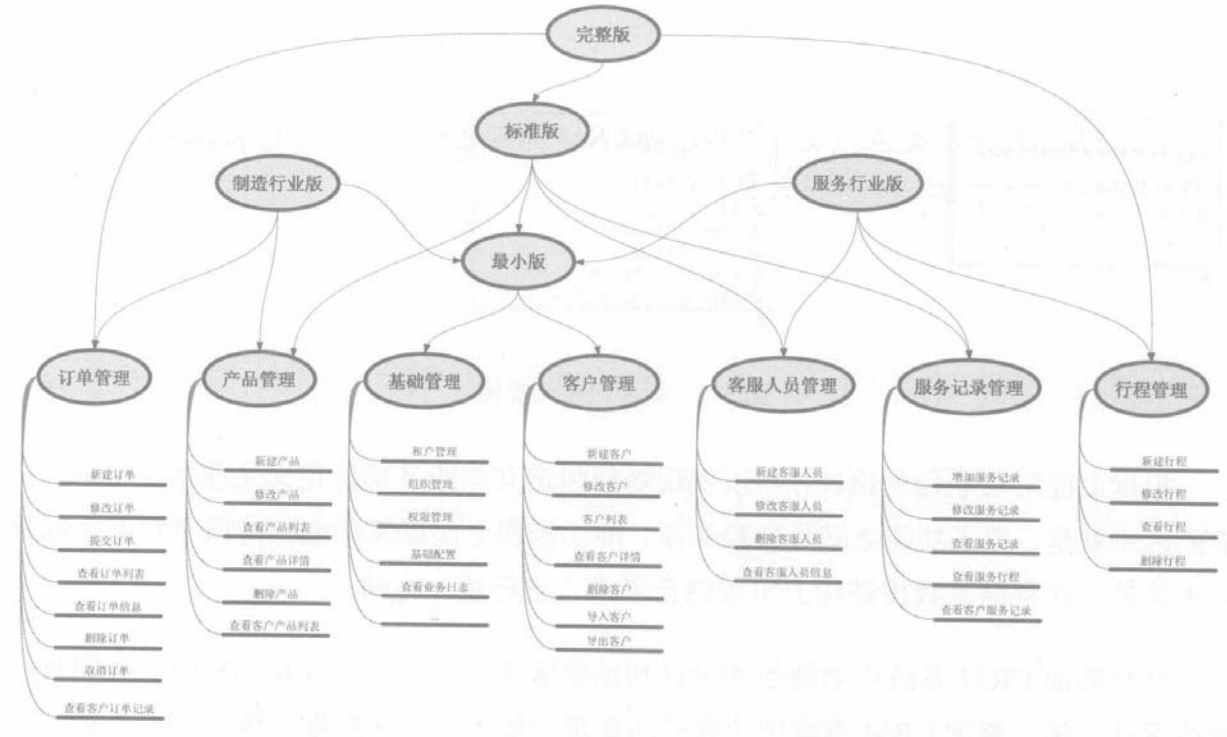


图 5-6　CRM 系统销售包设计示意图

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

3. 功能包定义

从前面的分析可以看出，功能包可以包含一组相关的原子功能，也有可能包括一组更小的功能包。

功能包定义，就是要对功能包进行描述，与原子功能定义类似，要定义相应的名称、关键字、内容等相关信息，及其所包括的原子功能集合和功能包的集合。有关功能包定义的设计可以参见图 5-7。

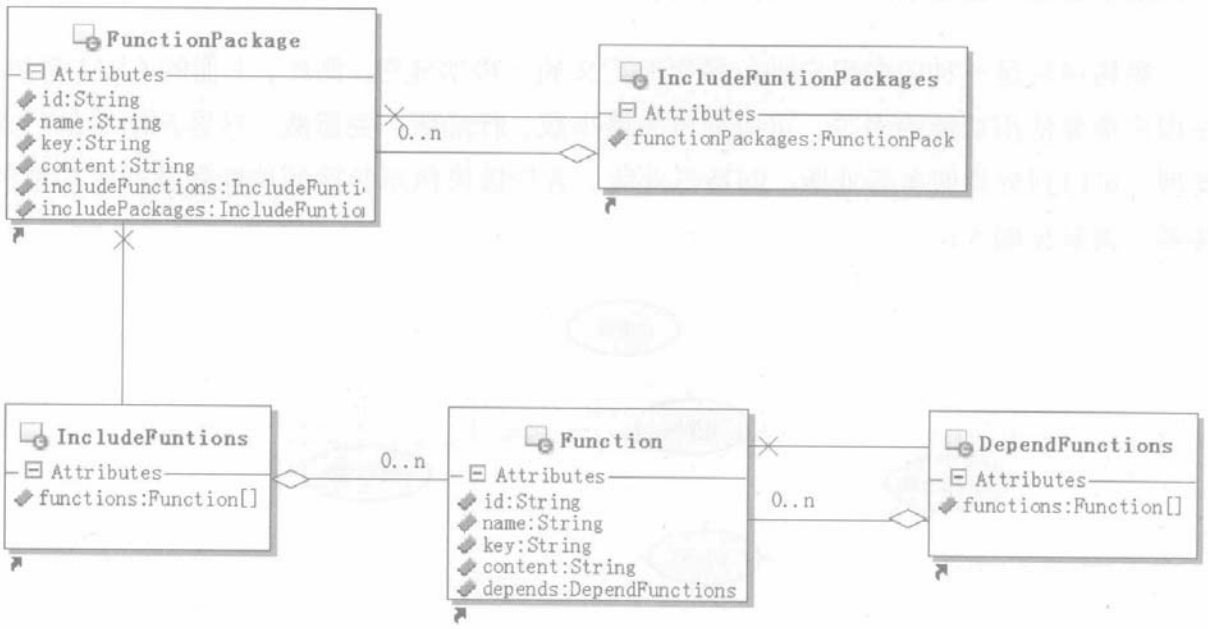
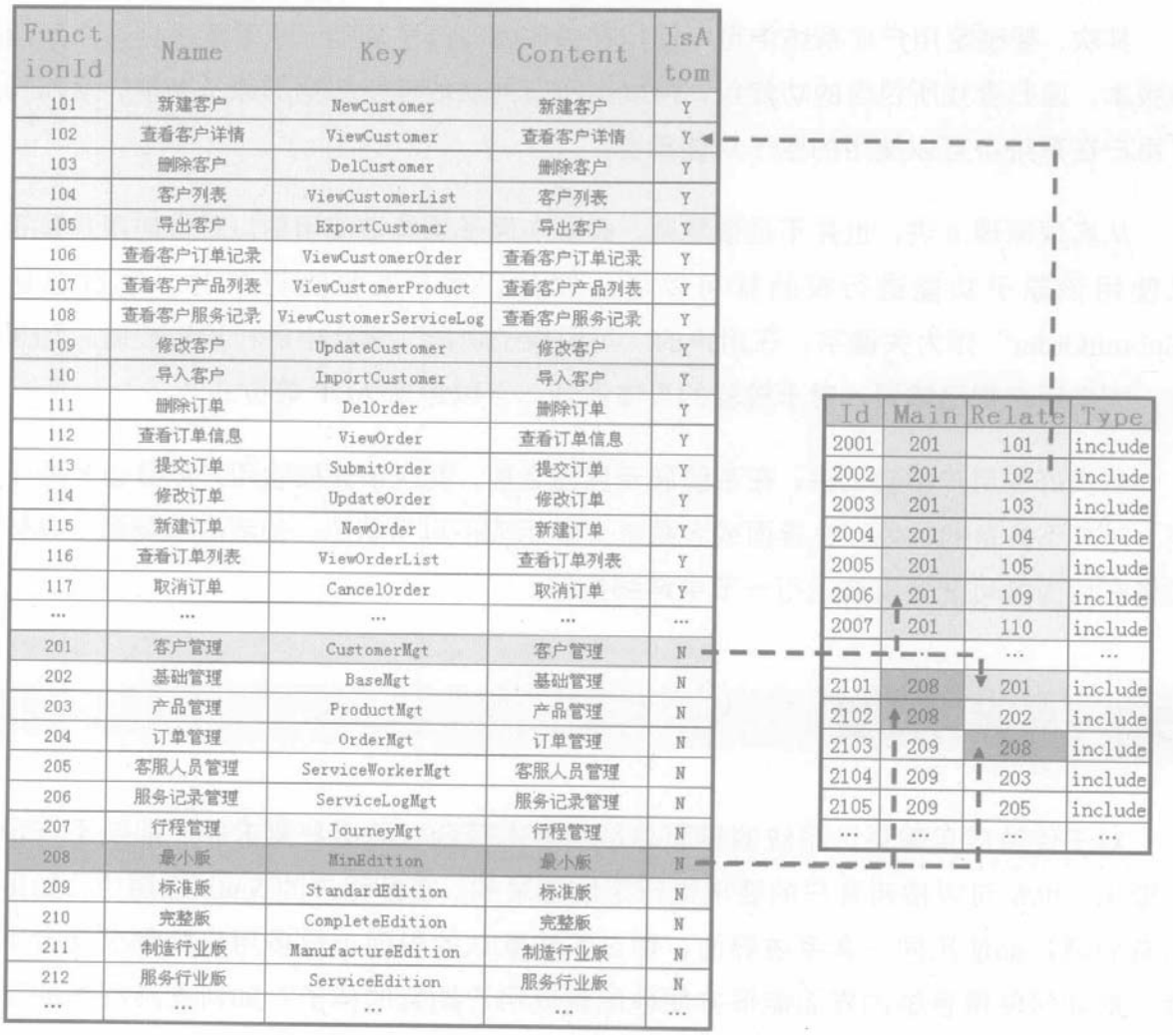


图 5-7　功能包定义类图

根据上面对功能包的设计，可以发现功能包定义与原子功能定义是非常类似的。二者的区别就是，原子功能之间是依赖关系，而功能包之间以及功能包与原子功能之间是包含关系。在存储的数据结构上可以综合考虑，进行统一存储。

结合前面 CRM 系统中功能包的设计和销售版本包的设计，以及功能包定义数据结构的设计，统一整理 CRM 系统中的原子功能及功能包的定义数据，请参见图 5-8。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



FunctionId	Name	Key	Content	IsAtom
101	新建客户	NewCustomer	新建客户	Y
102	查看客户详情	ViewCustomer	查看客户详情	Y
103	删除客户	DelCustomer	删除客户	Y
104	客户列表	ViewCustomerList	客户列表	Y
105	导出客户	ExportCustomer	导出客户	Y
106	查看客户订单记录	ViewCustomerOrder	查看客户订单记录	Y
107	查看客户产品列表	ViewCustomerProduct	查看客户产品列表	Y
108	查看客户服务记录	ViewCustomerServiceLog	查看客户服务记录	Y
109	修改客户	UpdateCustomer	修改客户	Y
110	导入客户	ImportCustomer	导入客户	Y
111	删除订单	DelOrder	删除订单	Y
112	查看订单信息	ViewOrder	查看订单信息	Y
113	提交订单	SubmitOrder	提交订单	Y
114	修改订单	UpdateOrder	修改订单	Y
115	新建订单	NewOrder	新建订单	Y
116	查看订单列表	ViewOrderList	查看订单列表	Y
117	取消订单	CancelOrder	取消订单	Y
...	...	...	...	...
201	客户管理	CustomerMgt	客户管理	N
202	基础管理	BaseMgt	基础管理	N
203	产品管理	ProductMgt	产品管理	N
204	订单管理	OrderMgt	订单管理	N
205	客服人员管理	ServiceWorkerMgt	客服人员管理	N
206	服务记录管理	ServiceLogMgt	服务记录管理	N
207	行程管理	JourneyMgt	行程管理	N
208	最小版	MinEdition	最小版	N
209	标准版	StandardEdition	标准版	N
210	完整版	CompleteEdition	完整版	N
211	制造行业版	ManufactureEdition	制造行业版	N
212	服务行业版	ServiceEdition	服务行业版	N
...	...	...	...	...

Id	Main	Relate	Type
2001	201	101	include
2002	201	102	include
2003	201	103	include
2004	201	104	include
2005	201	105	include
2006	201	109	include
2007	201	110	include
...	...	...	...
2101	208	201	include
2102	208	202	include
2103	209	208	include
2104	209	203	include
2105	209	205	include
...	...	...	...

图 5-8　CRM 系统原子功能及功能包结构示意图

5.2.3　功能使用校验

在经过对系统进行了原子功能划分和功能包的设计以后，系统的功能可以开始按照不同的版本进行销售了，但如何使得这种功能可配置的策略生效呢?

首先，不管用户购买的是什么版本或者什么功能包，系统都只针对原子功能进行校验。例如，在上面的 CRM 系统中，无论用户购买的是制造行业版，还是完整版，他都可以使用新建订单、修改订单、提交订单等与订单有关的功能。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

其次，要确定用户在系统中可以使用和操作哪些原子功能。这需要根据客户所购买的版本，递归查找所包含的功能包，再取出所有的功能包所包括的原子功能，从而构成了用户在系统中可以使用的原子功能集合。

从实现原理上讲，也并不是很复杂，只要在原子功能被使用前，对当前用户是否可以使用该原子功能进行校验就可以了。例如，客户在提交订单时，系统总是将“SubmitOrder”作为关键字，在用户的“可用原子功能”集合中进行检查校验。如果没有，则向用户提示错误。对于校验的具体实现，可以考虑 AOP 等方式。

从良好的用户体验上讲，在系统展示界面之前，就应该先校验用户所具备的原子功能。对其不具备的功能，在界面或者菜单上显示成不可用状态，或者直接隐藏。具体方案将在后面的可配置系统运行一节中详细探讨。

5.3 界面可配置

对于传统应用来说，系统的界面总是能基本符合当前用户要求的。即使不符合用户要求，也是可以按照客户的要求进行定制部署的。在多租户的 SaaS 应用中，如果让所有的用户都使用同一套系统界面，那么系统默认的界面与很多用户的需求就会不一致。那如何使得系统的界面能很方便地配置成用户期望的样子？如何支持界面的可配置？

界面可配置，主要包括两个方面：系统菜单可配置和页面内容可配置。

5.3.1 系统菜单可配置

前面讨论了功能的可配置，主要是指不同的租户可以同时购买和使用系统内不同的功能集合。而同样的功能菜单对于不同的租户来说，可能会有完全不一样的名字。例如，CRM 系统中的客户管理，在医院中使用，就得改成病人管理，客户服务人员就得改成医生，客户服务记录就是就诊记录等。这种菜单名称的配置及动态展示，是界面可配置要解决的一个问题。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

除了系统菜单的名字可配置以外，菜单的层次结构及分布，不同的租户可能也会有不同的要求，这也是需要可配置的。对于系统菜单的配置，在设计上还需要考虑以下几个问题：

- 一个租户一套菜单；
- 一个菜单可以关联一个原子功能；
- 组织成树状结构，构成上下级菜单结构；
- 同级菜单之间还存在显示顺序的问题。

综上所述，对系统菜单可配置内容进行简单的设计，可以得到类图，见图 5-9。

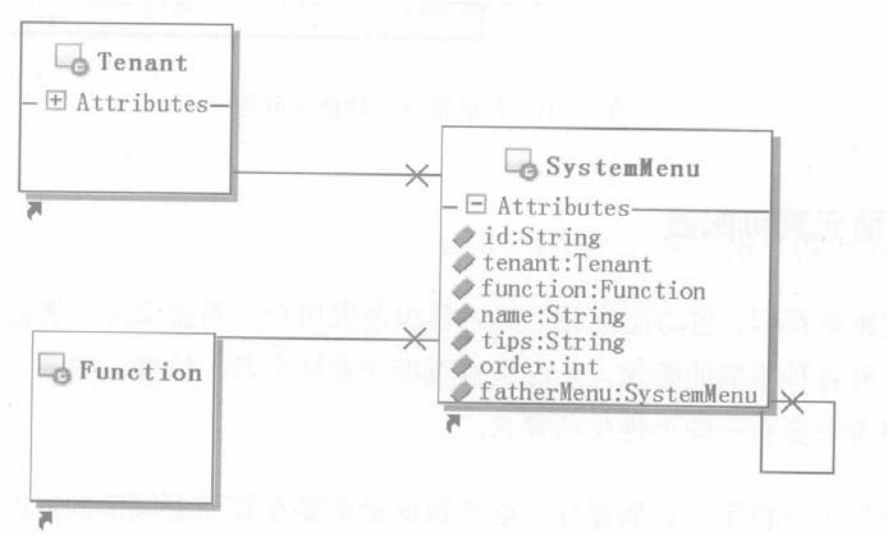
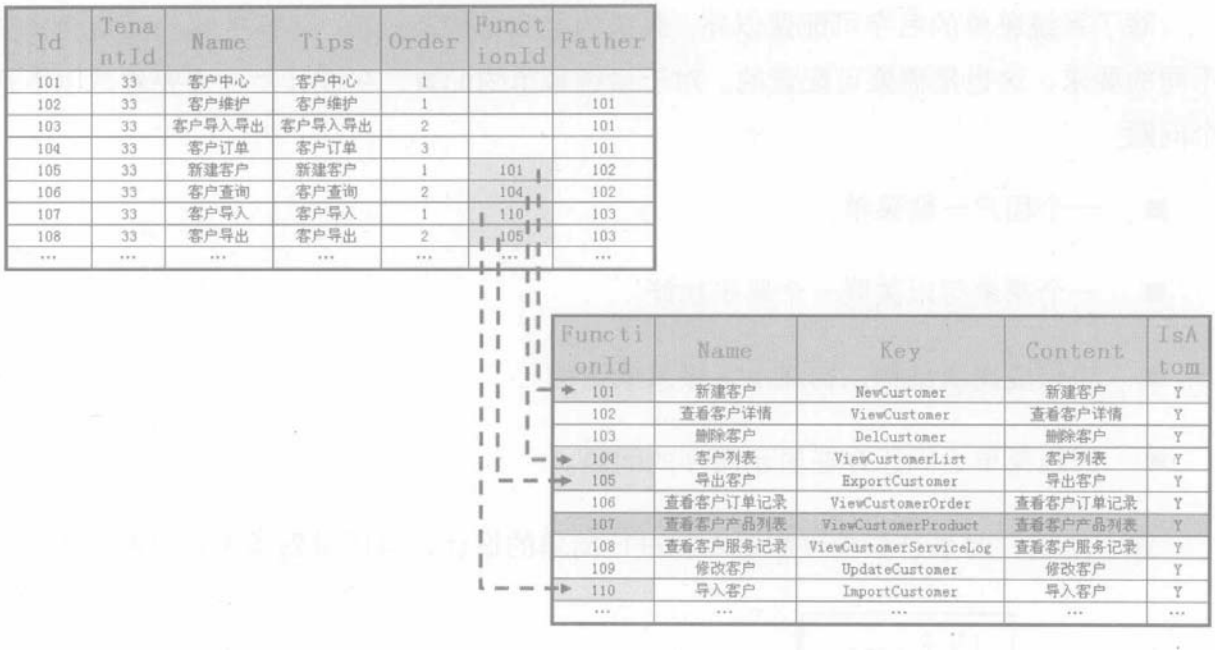


图 5-9　系统菜单配置类图

根据上面系统菜单配置类的设计，可以产生系统菜单与原子功能相关联的示例数据，见图 5-10。

本节主要介绍的是系统菜单的租户可配置性，有关系统菜单的加载显示和使用会在后面的章节中介绍。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



Id	TenantId	Name	Tips	Order	FunctionId	Father
101	33	客户中心	客户中心	1		
102	33	客户维护	客户维护	1		101
103	33	客户导入导出	客户导入导出	2		101
104	33	客户订单	客户订单	3		101
105	33	新建客户	新建客户	1	101	102
106	33	客户查询	客户查询	2	104	102
107	33	客户导入	客户导入	1	110	103
108	33	客户导出	客户导出	2	105	103
...	...	...	...	...	...	...

FunctionId	Name	Key	Content	IsAtom
101	新建客户	NewCustomer	新建客户	Y
102	查看客户详情	ViewCustomer	查看客户详情	Y
103	删除客户	DelCustomer	删除客户	Y
104	客户列表	ViewCustomerList	客户列表	Y
105	导出客户	ExportCustomer	导出客户	Y
106	查看客户订单记录	ViewCustomerOrder	查看客户订单记录	Y
107	查看客户产品列表	ViewCustomerProduct	查看客户产品列表	Y
108	查看客户服务记录	ViewCustomerServiceLog	查看客户服务记录	Y
109	修改客户	UpdateCustomer	修改客户	Y
110	导入客户	ImportCustomer	导入客户	Y
...	...	...	...	...

图 5-10　系统菜单与功能关联图

5.3.2　页面元素可配置

与系统菜单类似，各功能页面上的内容也是供用户与系统交互的界面元素。不同的租户可能会有各种不同的需求，无论是对页面元素的个数、位置、顺序，还是元素的含义，租户都可能会有一些个性化的需求。

由于租户可以自定义扩展数据，这些数据是需要在页面上展示的。因此，对于不同的租户，页面上元素个数可能是不一样的。同时，对于在设计时设定的页面元素，一般情况下是不允许删除的，但有时候还是允许租户将一些无关紧要的字段隐藏。

另外，同一个页面上同一个页面元素可能代表的含义是不一样的。例如，在“客户新建”页面有一个标签叫“客户姓名”，有的租户喜欢把它配置成“顾客姓名”，有的租户则更愿意配置成“代理商姓名”。另外，对于页面元素的位置，以及展示的顺序，不同的租户也会有不同的要求。对于功能页面上的元素进行可配置，将可以帮助租户更容易地理解和使用系统。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

页面内容可配置的需求和目标是比较明确的，但要实现得非常灵活和实用是一件很困难的事情。常见的做法是采用标签语言来描述，这种做法设计起来还是非常复杂的，对于进行页面配置的租户管理员要求是比较高的。在此不做过多的阐述。

5.4 流程可配置

流程可配置是工作流系统要解决的问题。在流程可配置方面，SaaS 应用与传统应用在本质上并没有太大的区别。只是在多租户的 SaaS 应用中，租户之间的流程和数据是要隔离的。除了应用预置的流程模板以外，其他的都是由租户自己来定义和设计的。

首先，工作流系统要能支持多租户的工作流设计。为了保证租户进行工作流程定义的简易性，系统应该提供公用的流程模板。租户可以复制这些公用流程模板，并在此基础上进行调整，以设计出适合于自身的工作流程。而租户设计出的流程总是属于租户独占的，其他租户是看不到的，租户之间的流程是隔离的。

其次，工作流引擎在运行时，总是能根据当前用户所属的租户，去查找属于该租户的流程，然后实例化流程。流程实例的流转与以往的工作流基本是一致的，但如果涉及组织和人员的选择，则应确保租户之间的数据隔离。

对于多租户的 SaaS 应用系统来说，工作流系统的设计和使用，除了上述差异以外，与传统应用系统没有本质上的区别，在此不做过多的阐述。

5.5 配置元数据管理

对于一个完整的 SaaS 应用来说，会有很多的功能模块都需要实现可配置的特性。如果各功能模块单独考虑可配置性的实现，对于系统业务逻辑的实现来说是非常复杂的。这样做不容易实现统一，用户体验会很差，功能业务的逻辑也会变得很复杂，在系统中实现可配置性的成本就会非常高。

首先，对于不同的业务模块来说，其可配置性的要求并没有差别。例如，在数据可

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

配置上，无论是对客户数据的配置，还是对产品数据的配置，如果采用名称值对方案的话，其实现是完全一样的。在功能可配置和界面可配置上更是统一的，各种业务功能的可配置性需求是大致相同，基本上一致的。对于这种通用逻辑的处理，如果分散到业务逻辑中处理，是不符合设计上的复用原则的。因此，对同一种类的多项配置可以进行统一实现。即脱离业务逻辑实现统一公共的数据可配置、功能可配置、界面可配置和流程可配置。

其次，前面介绍了数据、功能、界面、流程 4 个方面的可配置，分别介绍了 4 个方面的配置要求和配置的各种参数。从表面上看，这些参数是各不相同的，但是如果对其进行抽象的话，可以分析出四者之间是有很多共性的，至少在可配置参数的管理上是可能统一的。为此，可以考虑将数据可配置、功能可配置、界面可配置、流程可配置的配置参数进行抽象，对于系统内可配置的扩展点，以及租户基于这些扩展点进行配置的内容进行统一的管理。

最后，由于 4 种可配置的参数在元数据上得到了统一，那么，可以考虑提供一整套统一的公共的接口，用来使用和操作这些配置数据。

5.5.1 配置元数据

如何实现这些系统可配置参数的统一管理，在原理上可以参考 MDA（Model Driven Architecture，模型驱动架构）基于元数据管理的思想，可以将各种可配置参数通过元数据的方式进行统一。有关 MDA 思想的相关知识，可以参见下文：

> MDA（Model Driven Architecture）是模型驱动架构，它是由 OMG 定义的一个软件开发框架。它是一种基于 UML 以及其他工业标准的框架，支持软件设计和模型的可视化、存储和交换。和 UML 相比，MDA 能够创建出机器可读和高度抽象的模型，这些模型独立于实现技术，以标准化的方式储存。MDA 把建模语言用做一种编程语言，而不仅仅是设计语言。MDA 的关键之处是模型在软件开发中扮演了非常重要的角色。
>
> MDA 源自于众所周知的把系统操作的规范从系统利用底层平台能力的方式细节

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

中分离出来的思想，MDA 提供了一种途径（通过相关的工具）来规范化一个平台独立的系统、规范化平台、为系统选择一个特定的实现平台，并且把系统规范转换到特定的实现平台。MDA 的三个主要目标：通过架构性的分离来实现轻便性、互操作性和可重用性。

模型驱动架构（MDA）是 OMG 组织近年来一直热炒的一个新的技术体系，同时也是众多搞软件模型研究人员的一个新热点。MDA（模型驱动架构）核心的思路是希望通过对商业模型（比如企业信息化或建筑领域的解决方案）的领域研究，进而提炼出一个相对核心的领域模型，同时抽象出一个 PIM（平台无关模型）。之后根据不同的开发平台（例如.net 或 J2EE），应用平台（Windows 或 UNIX）形成相应的 PSM（平台相关模型）。

根据前面对数据可配置、功能可配置、界面可配置、流程可配置 4 种可配置内容的分析，可以总结出 SaaS 应用中要管理的可配置参数包括以下几种：表、原子功能、功能包、菜单、页面、页面元素、流程等。对于以上这些配置类型，暂且称之为配置元。前面讨论的所有可配置内容都可以通过以上这些配置元来表示。

对于这些配置元，可以采取 MDA 的思想进行抽象，可以得到这些配置元的通用数据模型，以及通用数据以外的扩展数据。

除了配置元的数据定义以外，还涉及配置元之间的关系。根据前面的分析，可以总结出系统中包含以下几种关系：

- 原子功能之间的依赖；
- 功能包对原子功能的包含；
- 功能包对功能包的包含；
- 菜单和菜单的上下级包含关系；
- 菜单到功能之间的关联关系；

SaaS 架构设计
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 页面对页面元素的包含关系。

配置元之间存在三种关系类型：包含、依赖、关联。根据上面的抽象，可以得到配置元模型，模型图可以参见图 5-11。

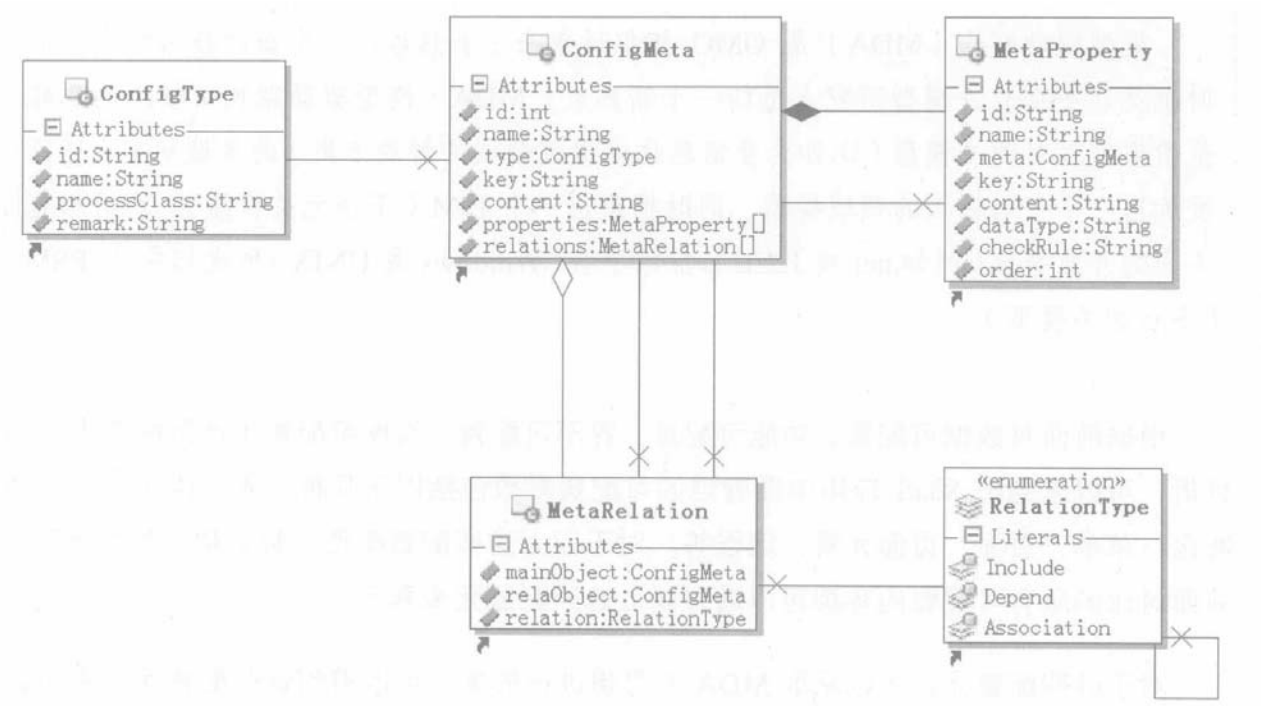


图 5-11　配置元数据类图

其中 ConfigType 就是上面的配置元类别，包括表、原子功能、功能包、菜单、页面、页面元素、流程等。每个类别对应一个相应的处理类，由这个类来负责相应配置元数据的处理。ConfigMeta 用来表示配置元的公共属性，对配置元的扩展属性，通过 MetaProperty 来描述，每一项 MetaProperty 用来表示配置元的一个扩展属性。每个配置元还包括一个与其相关的配置元的集合。

根据上面的配置元模型，可以定义出元数据的结构。元数据的存储，可以采用文件存储，也可以考虑采用结构化数据库存储。以 CRM 系统为例，其配置元数据包含类似图 5-12 中的内容。其中，相互关联内容，可以参照表格的颜色和箭头指向关联。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

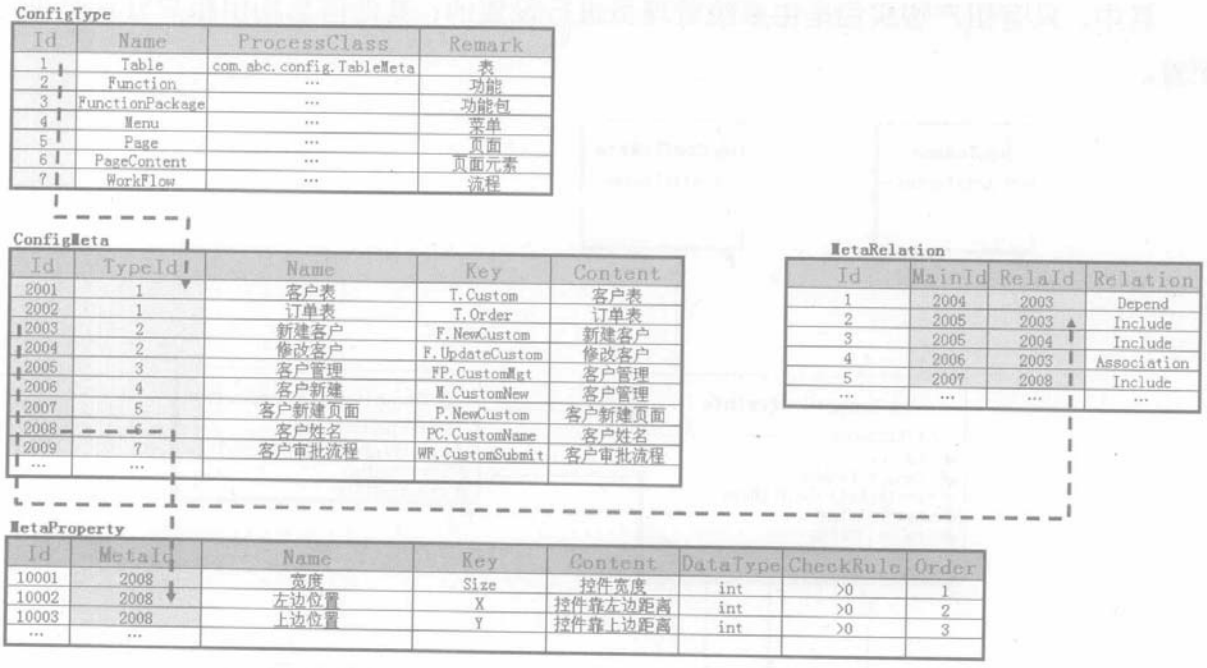


图 5-12　配置元数据示例图

配置元数据是与系统内的设计密切相关的。必须由 SaaS 应用的开发者根据系统的规划和设计，来确定系统内可配置点，才能定义出配置元数据的内容。

5.5.2　租户配置数据

租户配置数据，是指基于配置元模型和配置元数据，由租户管理员根据租户的需要来定义相应的配置信息。租户配置数据的设计模型可以参见图 5-13。

与租户相关的配置信息包括：

- 数据表扩展字段；
- 租户购买包；
- 自定义菜单；
- 自定义页面元素。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

其中，只有租户购买包是由系统管理员进行设置的，其他信息均由租户管理员进行配置。

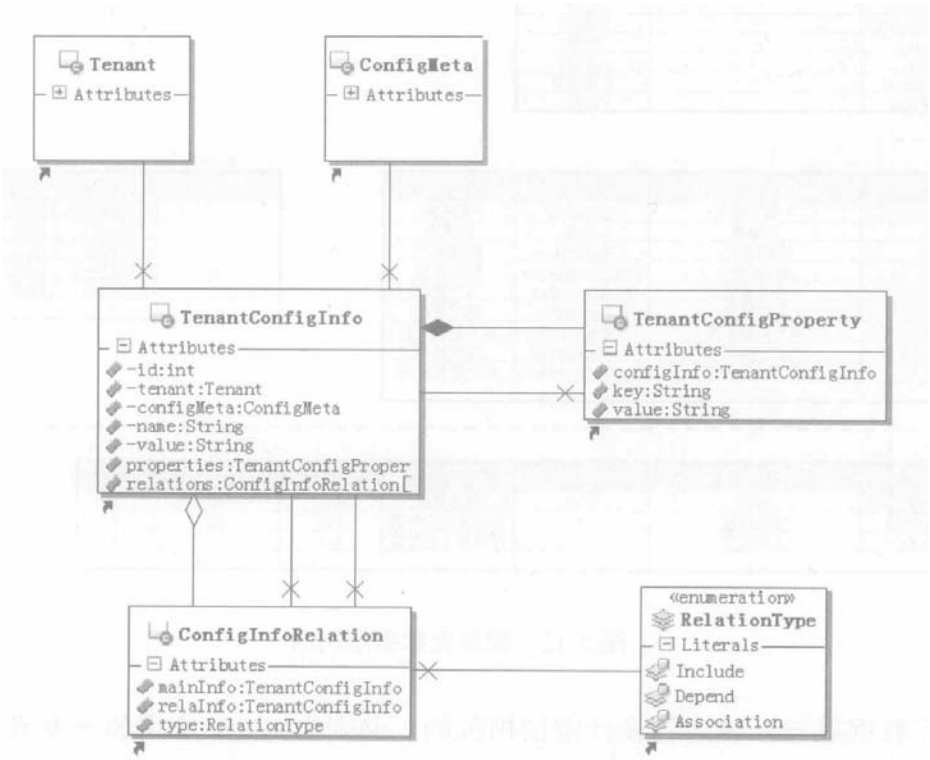


图 5-13　租户配置数据类图

根据上面的租户配置数据设计，可以定义出租户进行配置生成的数据结构。租户配置数据可以基于文件或结构化数据库来管理，数据示例可以参见图 5-14。

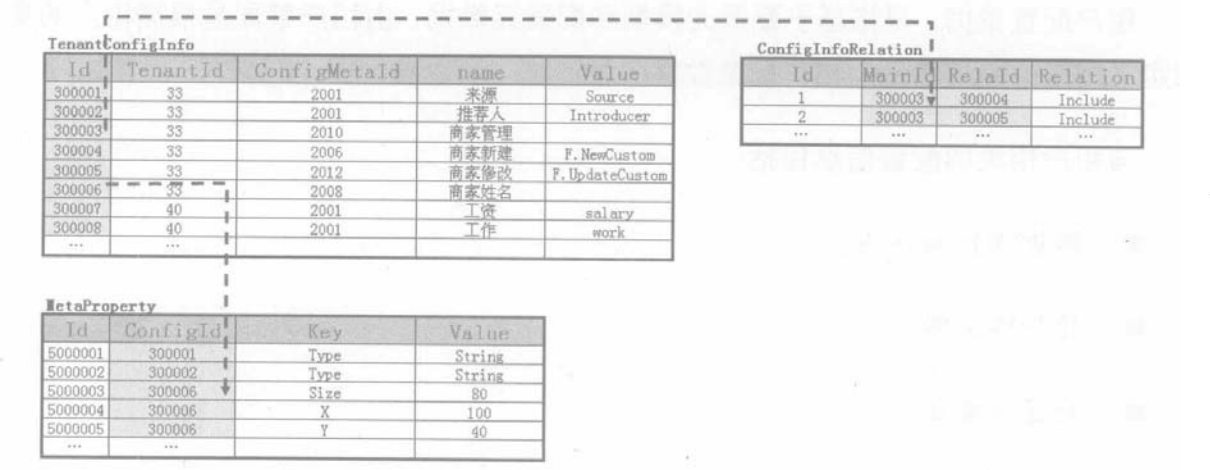


图 5-14　租户配置数据示例图

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

5.5.3 配置元数据服务

通过前面的配置元数据管理和租户配置数据管理，就可以将系统内可以配置的元素，以及各租户所做的相应配置数据定义出来。那么，系统中的各业务模块该如何来使用这些元数据呢？这就需要实现公共的配置元数据服务。

配置元数据服务，就是要提供一套统一操作这些元数据的服务接口，以及一套可以直接使用元数据的控件，供业务模块开发时使用。这样就可以简化业务功能实现时实现可配置性的复杂度，同时也保证了统一的配置规划和管理风格。

5.6 可配置系统运行

前面已经对多租户的 SaaS 应用在数据、功能、界面和流程 4 个方面要解决的可配置问题，以及配置元数据相关管理进行了详细的阐述，那么这些可配置内容如何在系统中发生作用呢？系统如何基于这些配置数据运行呢？

首先，正如前一节中所讲的元数据服务，为整个系统操作和使用元数据提供一套统一的服务接口，由于最终租户在使用时需要配置这些元数据，所以系统需要为租户提供一套租户配置界面，让租户可以很轻松地配置和修改这些数据。

其次，所有这些配置数据要在系统中生效的话，需要在系统界面上体现。主要包括两个方面：一方面是系统菜单的体现，另一方面是功能页面上的体现。系统菜单的体现通过系统菜单框架来实现，而功能页面上各配置要素的体现，需要通过功能页面容器来实现。

最后，有关各类配置数据的使用逻辑及调度控制，通过 3 个特殊的引擎来完成。流程可配置数据的使用与控制，由工作流引擎来实现。有关扩展数据的查询、使用、提交及相应的检查等，由专门的扩展数据引擎来完成。设计单独的功能引擎，负责系统内功能的调度和租户对功能的使用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

因此，可配置系统运行，需要包括元数据服务、租户配置 UI、系统菜单框架、功能页面容器、功能引擎、扩展数据引擎、工作流引擎等配合实现，还包括配置数据和扩展数据两部分。具体关系见图 5-15。

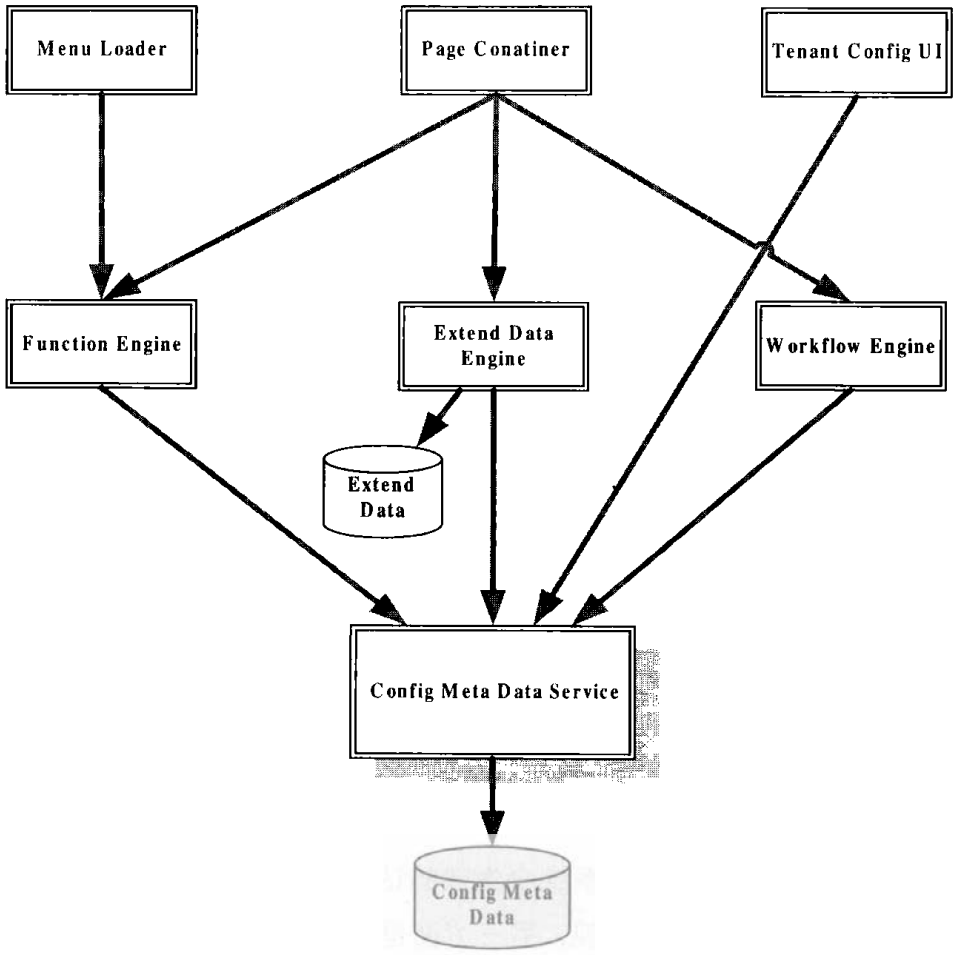


图 5-15　可配置系统运行结构图

下面对其中几个主要的部分分别进行介绍。

5.6.1　系统菜单框架

系统菜单框架在现有的传统应用中一直都是存在的，主要负责系统功能菜单的展示。与传统应用相比，SaaS 应用的各个租户可以对系统菜单进行各自的定制，因此，系

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

统菜单框架需要根据租户所配置的菜单进行装载。

另外，由于 SaaS 应用支持租户“按需使用，按需付费”，各租户所购买和使用的功能是不一致的。因此，系统菜单框架需要依赖功能引擎来装载租户菜单所对应的功能，同时，在菜单被单击使用的时候，需要通过功能引擎来实现对原子功能的调度和使用。

5.6.2 功能页面容器

功能页面容器主要解决页面要素展示、页面功能可用性控制、可配置数据展示、可配置数据提交 4 个方面的问题。

页面要素展示就是要根据系统页面的设计和租户的配置，来展示页面上的各种展示要素，如果要允许租户做很复杂的配置，并要支持页面丰富多彩的展示，需要使用较复杂的标签语言来实现。

页面功能可用性控制，就是根据租户购买的原子功能列表，来确定页面内的功能是否可用。对于不可用的功能，显示成不可用状态，或者隐藏。

可配置数据展示，就是根据租户针对页面实体定义可配置数据列表，在页面内自动展示。租户配置的实体，对于当前界面来说，可能是主要实体，也可能是其中一个子实体。也就是说，给这个实体配置的数据项，可能是界面上的单个数据元素，也可能是界面中某个集合数据元素的数据列。对于单个数据项，可以定制一个通用控件根据配置的数据类型来动态显示。对于列表数据项，则需要通过定制特殊的表格控件来实现。

可配置数据提交，就是根据租户针对页面实体设计的可配置数据项，当页面提交时，从界面上收集可配置项的数据内容，与数据实体一并提交。如果涉及可配置数据的校验，可以采用正则表达式的方式定义在租户元数据中，并在提交前进行校验。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

5.6.3 扩展数据引擎

数据可配置性的实现，主要包括扩展数据的展示、扩展数据的提交、扩展数据的校验与保存。扩展数据的展示，就是根据租户对扩展数据的配置，将扩展数据进行类型化转换以后，在功能页面中展示出来。而扩展数据的提交，就是通过功能页面收集租户所配置的扩展数据项相对应的数据值，能够按扩展数据的实体类型进行提交。扩展数据的校验与保存就是对提交过来的扩展数据，根据租户配置的要求，进行类型和数值上的校验，校验通过后，保存到扩展数据库。

有关扩展数据的展示和提交，都需要依赖功能页面容器来实现。

5.6.4 功能引擎

功能引擎，就是要根据租户所购买的功能包，来查找租户所有的“可用原子功能”集合，结合其购买的期限，来确定当前租户是否可执行相应的功能或操作，以及实现对相应原子功能的调度。

功能可配置的实现，最终需要依赖系统菜单框架和功能页面容器两方面来实现。由系统菜单作为入口的，由系统菜单框架负责控制；在功能页面内，提供相应业务功能的，由功能页面容器负责控制。

5.7 小结

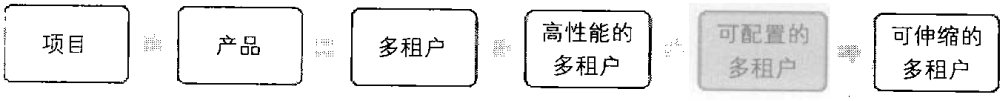
本章主要针对 SaaS 应用的可配置性，从数据、功能、界面、流程 4 个方面，阐述了 SaaS 应用如何提供很好的可配置性，来适应各种不同类型的租户的实际情况，并满足用户的各种定制化需要。重点讨论了数据可配置和功能可配置的实现方案。

另外，对于可配置性的实现方式，在 SaaS 应用中可以实现配置元数据服务，实现统一的有关配置信息的管理和使用。通过由应用开发者来设计系统中可以配置的系统配

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

置元数据，由租户管理员在统一的界面中管理租户配置元数据。整个系统基于公共的元数据服务来运行。

最后，简单介绍了在 SaaS 应用中如何实现可配置系统的运行。实现灵活、方便的可配置性是 SaaS 应用能否满足大量租户各种定制化需求的一个重要基础。只有支持租户对系统的可配置，才能满足大量租户的特定需求，占领市场。这也是 SaaS 应用成熟的一个重要标志。



Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第6章 可伸缩的SaaS应用架构

6.1 伸缩性（Scalable）的概念……132
6.2 应用服务器层的水平扩展……133
6.2.1 基于Session复制的水平扩展方式……134
6.2.2 基于Session Sticky的水平扩展方式……137
6.2.3 基于Cache的集中式Session实现水平扩展……138
6.2.4 三种水平扩展方式的比较……139
6.3 数据库层的水平扩展……139
6.3.1 数据库的垂直切分……140
6.3.2 数据库的读/写分离技术……141
6.3.3 数据库的水平切分……142
6.3.4 三种数据库层的水平扩展方案对比……145
6.4 小结……146

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

随着定制化功能的完善，侠客录 CRM 的用户数进一步稳定增长。2000 个用户的目标已经唾手可得，接下来就得朝着 3000 个用户的目标进发了。

3000 × 50 × 12 = 180 万/年，郭靖已经在计算每年的收入了。看样子最近系统比较稳定，郭靖也比较清闲，开始关注未来的市场了。

杨康：郭靖，最近国内电子商务巨头阿里巴巴旗下的阿里软件公司推出了开放式的 SaaS 平台，说要和我们这些小的软件开发商共同打造一个 SaaS 生态圈。你看看，这是他们的宣传彩页，我们是不是也有机会跟他们合作。他们既有大的品牌，又有大量的中小企业客户，跟他们合作对我们的市场拓展可是非常好呀。

郭靖：阿里巴巴？我前段时间还听说他们也进军 SaaS 行业，当时还以为是我们的竞争对手呢。这样看起来，也许我们还真的可以靠上这棵大树呀。

杨康：嗯，那你先上他们网站看看详细的技术资料吧。据说技术改造是相当简单的，几个小时就够了。我们的软件基本不用怎么改，就可以把他们的客户拉过来用我们的 CRM。我这就去跟他们的商务部门联系一下，看能否和他们开展合作。

果然，技术改造的过程相当简单。了解他们平台的原理之后，郭靖花了两个小时，就把侠客录 CRM 改造成可以接入阿里软件互联平台的应用了。接下来就只等杨康把商务方面谈妥，就可以在阿里的平台上架销售了。

杨康动作也很快，很快就搞定了商务方面的合作。

阿里软件互联平台 ISV 合作流程见图 6-1。

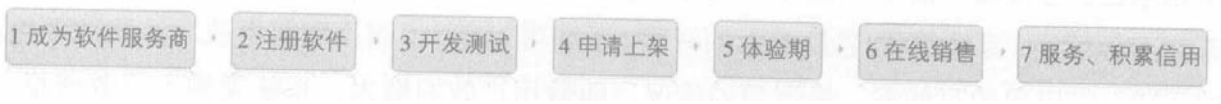


图 6-1　阿里软件互联平台 ISV 合作流程

杨康：阿里巴巴对我们的软件很满意，高度赞赏了我们的响应速度。真不敢相信，几个小时就接入平台了，你真牛！

杨康：不过，阿里巴巴也提出了一点担心。我们的软件做得不错，可能会受到用户

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

的广泛欢迎。但是，我们的系统能支撑 10 万级的用户数吗？在阿里巴巴平台上备受欢迎的软件有很多用户数都已经达到了 10 万级别，并且增长速度还很快。

郭靖：10 万级呀？我们现在才 2000 不到呀！我还以为今年年底能到 3000 就不错了。那么，如果接入阿里的平台，估计要多久才能达到 10 万呢？

杨康：如果运营推广得好的话，也挺快的。因为先期也还是要免费试用的，所以半年就可能达到 10 万了。

郭靖：那再给我三个月时间，我要把我们的架构再做一次大的调整，以适应以 10 万计的用户量。当然现在先上阿里巴巴的平台也没关系，但是用户数先控制一下，先期不要做大规模推广。

杨康：OK！前三个月我们可以先控制一下用户数，这样还可以造成稀缺效应。不过三个月后，我可是要大规模运营了，你可要撑得住呀。

郭靖：No Problem!

6.1 伸缩性（Scalable）的概念

郭靖的自信来源于他对“Scalable”的理解。“Scalable”很多时候也翻译为“可扩展”。不过，翻译为“可扩展”容易被误解为功能可扩展（Extendable），而“Scalable”更多强调的是性能、容量等方面的可扩展。因此，将“Scalable”翻译为“可伸缩”会显得更恰当些。可伸缩字面上也很容易理解，即，系统可伸可缩。系统规模/容量较小的时候，系统可以缩小；系统规模/容量增大的时候，系统可以增大，而不需要更改系统整体架构。对于 SaaS 应用的可伸缩，最理想的情况：随着用户数的增大，系统架构不用做调整，而仅需要增加/增强相应的硬件设备（应用服务器、数据库服务器）即可。

实现可伸缩的最简单方式就是“Scale up”。“Scale up”通常翻译为垂直扩展或者向上扩展（这个地方没办法，翻译成垂直伸缩或者向上伸缩就太别扭了），也就是增强硬件设备。这种方式几乎是任何应用架构普遍适用的，但是通常都会面临高成本的问题。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

比如郭靖这样的小公司是不太可能承受 IBM 小型机这样的成本的，尽管小型机可能在性能、处理能力上能够达到普通 PC Server 的几倍到几十倍。

而通常强调的应用架构具有可伸缩性，一般指的是可以实现“Scale out”，即水平扩展或者向外扩展。实际上目前世界上最大的搜索引擎 Google 就是这一方面的典范。Google 的搜索引擎需要强大的计算能力，但是其整体架构是由数千、数万台普通的 PC Server 构建的，而不是依赖于单个的强大的大型服务器。这样，就能够确保随着互联网信息的日益丰富，Google 搜索引擎并不会因为其搜索的信息量越来越大而导致无法扩展的问题。

对于郭靖的侠客录 CRM 系统而言，只要实现了真正意义上的水平扩展，以目前 2 台 PC Server（1 台应用服务器+1 台数据库服务器）即能支撑 3000～5000 个用户的性能看，要达到 10 万的用户量，只是需要再增加 40 台左右 PC Server 即可。相对于带来的收益（10 万×50×12=6000 万/年），40 台 PC Server 的成本显然是很小的。

6.2 应用服务器层的水平扩展

实现应用服务器层的负载均衡，是实现应用服务器层水平扩展的最主要手段。具体实现负载均衡的策略有如下两种：

- 基于硬件负载均衡设备实现负载均衡，例如 F5 设备。
- 基于软件的方式实现负载均衡，例如通过配置 Apache Http Server。

尽管使用硬件负载均衡设备在性能等各方面具有更好的优势，但是对于郭靖这样的小公司来说，还没有能力也没有必要付出这样的成本。因此，对于郭靖来说，首先就是通过 Apache 来实现负载均衡。此时，郭靖设计的应用架构如图 6-2 所示。

Apache 可以实现负载均衡，根据服务器的压力情况，将每个用户请求分布到不同的应用服务器上。但是，大部分应用的用户请求可能是有状态的（一般使用 Session 记录用户状态）。这种情况下，如何能够在多台应用服务器之间保持用户状态，将是实现

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

应用服务器层水平扩展的关键。在处理这个问题上，通常的策略有如下三种：

- Session 复制；
- Session Sticky；
- 基于 Cache 的集中式 Session。

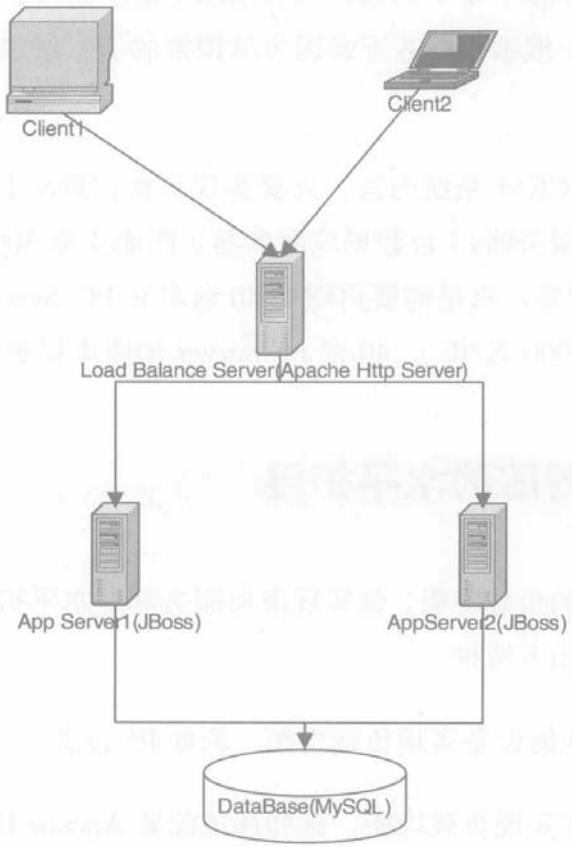


图 6-2　通过 Apache 实现应用服务器层的负载均衡

接下来的三节中我们将对这三种策略做具体的描述，并进行综合比较。

6.2.1　基于 Session 复制的水平扩展方式

很多应用服务器都强调支持强大的集群功能，Session 复制就是强大的集群功能特

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

性之一。通过 Session 复制，大部分应用都可以实现应用服务器集群。通过增加应用服务器集群中的服务器数量，应用就可以达到水平扩展的目的。

接下来我们将以具体的 Apache + JBoss 为例，配置 Load Balance 和 Session 复制。

首先，我们需要配置 Apache 实现 Load Balance。

1. 安装并加载 jk 模块

http.conf 文件添加如下内容：

```
include "conf/mod_jk.conf"
```

2. 配置 mod_jk.conf 文件

```
#加载 mod_jk Module
LoadModule jk_module modules/mod_jk-apache-2.0.55.so
#指定 workers.properties 文件路径
JkWorkersFile conf/workers.properties
#指定那些请求交给 JBoss 处理,"loadbalancer"为在 workers.propertise 里指定的负载分配控制器
JkMount /* loadbalancer
```

3. 配置 workers.properties 文件

```
worker.list = controller,jboss1,jboss2  #Server 列表
#========jboss1========
worker.jboss1.port=8009  #ajp13 端口号,在 Tomcat 下 server.xml 配置,默认 8009
worker.jboss1.host=192.168.0.225  #应用服务器地址
worker.jboss1.type=ajp13
worker.jboss1.lbfactor = 1   #Server 的加权比重，值越高，分得的请求越多

#========jboss2========
worker.jboss2.port=8009  #ajp13 端口号,在 Tomcat 下 server.xml 配置,默认 8009
worker.jboss2.host=192.168.0.226  #应用服务器地址
worker.jboss.type=ajp13
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
worker.jboss.lbfactor = 1    #Server 的加权比重，值越高，分得的请求越多

#========loadbalancer,负载均衡控制器========
worker. loadbalancer.type=lb
worker. loadbalancer.balanced_workers=jboss1,jboss2 #指定分担请求的 JBoss
worker. loadbalancer.sticky_session=0 #这里暂不配置 Session  Sticky
```

经过 Load Balance 的配置，应用层的压力就会均摊到多台应用服务器上，确保了应用层的压力得到有效的缓解。后续压力进一步加大时，也可以通过增加应用服务器，进一步分摊压力来扩展，确保了应用层具有良好的伸缩性。

但是，大部分的请求都是有用户状态的。单纯依赖上述配置，这个应用服务器集群中的每台服务器之间都是彼此独立的。如果一个用户的两次请求被 Load Balance 设备分发到两台不同的应用服务器上，则有可能会出现 Session 丢失的情况发生。Session 复制技术就是在这样的背景下产生的。

接下来我们将以 JBoss 为例，描述 Session 复制的具体配置方式。

（1）JBoss Cluster 作为一个 sar 包含在 JBoss 的$JBOSS_HOME/server/all/deploy/jboss-web-cluster.sar 目录下，要启用 Cluster 的话，需要启动 all 这个 Server。

（2）修改 jboss-web-cluster.sar/META-INF/jboss-service.xml 配置文件，配置相应的参数，包括使用 udp 还是 tcp 方式同步 Session，以及集群名称、集群中的服务器 IP 和端口等。

配置为 Session 复制之后，无论前端 Apache 怎样分发请求，都能够保证每个用户在这个集群的任何一台服务器上都具有相同的 Session，从而实现负载均衡。

Session 复制的实现技术非常复杂，在大规模集群中实用性并不强。在大规模集群中，服务器之间大量的 Session 复制会严重影响这些服务器的性能。而且随着服务器数量的增加，这种性能影响会显得更加突出，甚至不可接受。可见，在大部分互联网应用中，Session 复制技术应该是很少被采用的。

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

6.2.2 基于 Session Sticky 的水平扩展方式

为了避免 Session 复制所带来的性能影响，更简单、也是更高效的一种做法是 Session Sticky。这种方式将同一用户的请求转发到特定的 JBoss 服务器上，避免了集群中 Session 的复制。

基于 Session Sticky 实现应用层的水平扩展的部署模型如图 6-3 所示。

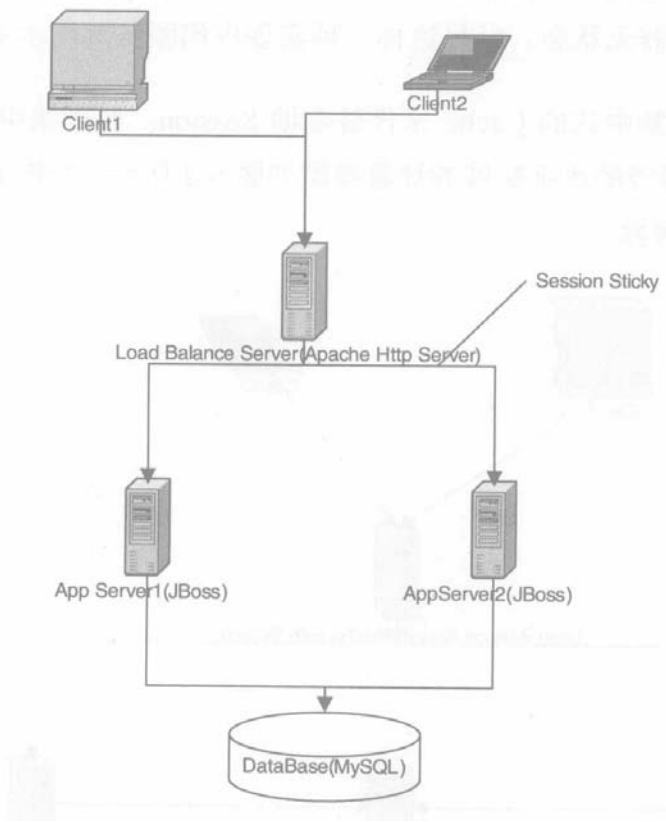


图 6-3 基于 Session Sticky 实现应用层的水平扩展的部署模型

这种方式与上面的 Apache 配置相比有细小的调整，即将 worker.properties 文件中的 worker. loadbalancer.sticky_session 由 0 调整为 1.而在 JBoss 上面，则不需要做任何集群的配置。

Session Sticky 的实现确实非常简单，不过这种方式不能满足 fail-over 的需求。即当

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

一台应用服务器“down”的时候，这台服务器上正在访问的所有用户的 Session 都失效了，所有用户不得不再次重新登录。此外由于 Apache 要基于 Session 而不是 Request 来分发请求，这种方式还容易导致负载不够均衡。

6.2.3 基于 Cache 的集中式 Session 实现水平扩展

Session 复制的方式性能较差，而 Session Sticky 的方式又无法保证 fail-over。那么，有没有一种方式能够克服这两种缺点呢？将 Session 集中到一台/一组独立的 Session 服务器上，确保应用服务器无状态，就是这样一种实现应用服务器层水平扩展的方法。

这种方案通常使用集中式的 Cache 来代替本地 Session。基于集中式 Cache 来代替 Session 实现应用服务器层的水平扩展的部署模型如图 6-4 所示（与图 6-2 相比，多了一个集中式的 Session 服务器）。

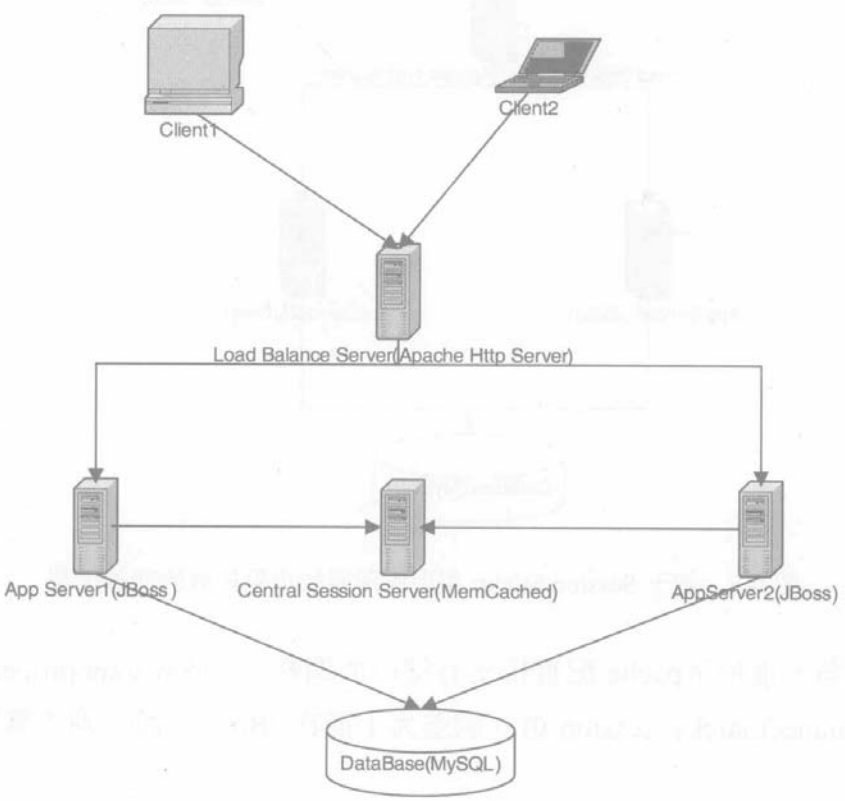


图 6-4 基于 Cache 的集中式 Session 实现应用层的水平扩展的部署模型

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

这样，应用服务器层基本上实现了完全的水平扩展，应用服务器层的压力基本上可以完全通过增加服务器进行扩展了。而集中式的 Session 服务器采用的是 MemCached，其本身也具备水平扩展能力。当 Session 数量大到一台 Cache 服务器都无法承受的程度时，我们也仅需要增加相应的 Cache 服务器即可，应用整体架构不会发生变化。

6.2.4 三种水平扩展方式的比较

前面其实已经对三种方式的优缺点做了详细的描述，下面再以表格的形式简单比较一下三种方案（见表 6-1）。

表 6-1 实现应用服务器层水平扩展的三种方式比较

实现方式	优势	劣势
Session 复制	■ 服务器负载可以得到较好的均衡，也可以确保 fail-over 的支持	■ Session 复制会对服务器端网络环境带来巨大的压力，尤其在应用服务器数量较大的时候，基本不适用于大型互联网应用 ■ 需要相应的应用服务器支持
Session Sticky	■ 实现比较简单，在 Load Balance 层做相应的配置即可 ■ 不会带来 Session 复制所引起的网络环境的压力	■ 不能实现完全的负载均衡，部分情况下负载会极端失衡 ■ 无法实现 fail-over
基于 Cache 的集中式 Session	■ 应用服务器层无状态，可实现完全的负载均衡 ■ 不会带来 Session 复制所引起的网络环境的压力	■ 实现相对复杂一些 ■ Cache 本身可靠性不能绝对保证，可能会造成部分 Session 的丢失

6.3 数据库层的水平扩展

相对于应用服务器层的水平扩展，数据库层的水平扩展更难实现。因此，很多应用在伸缩性上最后面临的问题大多是，数据库成为系统性能的瓶颈。应该说，实现数据库层的水平扩展，是实现 SaaS 成熟度模型第四级所面临的最大挑战。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

实现数据库层的水平扩展也有多种方式：

- **数据库的垂直切分**：将不同的功能模块所涉及的表划分到不同的物理数据库中，从而将对这些表的访问压力分担到多个不同的物理数据库中。
- **数据库的读/写分离**：同一个数据库在多个物理服务器上具有多份 Copy，彼此同步。然后将对于数据库的写操作都统一到一个主服务器上，而读操作则分摊到多台从服务器上。通过读/写分离，实现数据库访问压力的分担。
- **数据库的水平切分**：将原来存储在一个数据表中的数据，按照一定的规则，切分到多个不同的物理数据库中。每个数据库的数据结构完全相同，但是数据各不相同。最终对于业务数据的访问，会根据其数据所在的数据库，定位到某一个数据库中查询。

接下来我们将详细描述这三种数据库层的水平扩展方式。

6.3.1 数据库的垂直切分

数据库服务器成为系统性能瓶颈之后，要实现其水平扩展，首先最容易想到的就是数据库的垂直切分。即，将集中式的数据库表按照功能模块划分到不同的物理数据库中。例如，侠客录 CRM 中，既有客户管理相关的功能模块，也有论坛等与核心业务关联不大的模块。对于论坛这样的与其他模块没有紧密耦合的功能模块，其涉及的数据库表就是首先可以垂直切分的。

数据库层垂直切分后的系统部署模型如图 6-5 所示。

尽管数据库的垂直切分是最容易想到的。但是，对于大部分应用而言，除非模块间的关联很少，否则要实现垂直切分也不容易：

（1）原本可能存在的表连接，需要想办法去除；

（2）原本同一个数据库的事务操作，可能会变成跨数据库的事务。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

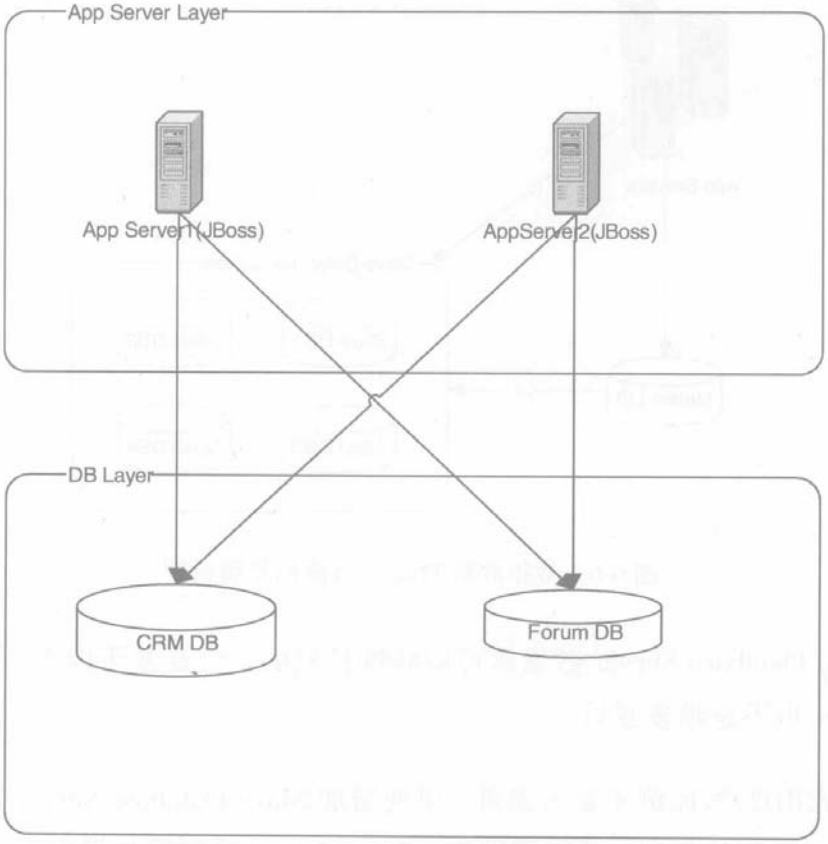


图 6-5　数据库层垂直切分后的系统部署模型

可见，数据库的垂直切分是一个可以适当采用，但是很难广泛采用的数据库层扩展技术。数据库垂直切分到一定粒度以后，就很难继续无限制地切分下去，或者进一步切分可能付出非常大的系统改造代价，甚至根本不可能。

6.3.2　数据库的读/写分离技术

对于读多写少的互联网应用，会广泛采用读/写分离技术。例如，MySQL 的 Replication 技术就是被广泛使用的读/写分离技术。虽然，SaaS 应用读多写少的特征不是特别明显，但是，读对于数据库的压力占数据库总负载的比例还是不小。因此，读/写分离也是提升 SaaS 应用数据库性能，实现数据库层扩展性的一个重要方法。

采用读/写分离的应用架构如图 6-6 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

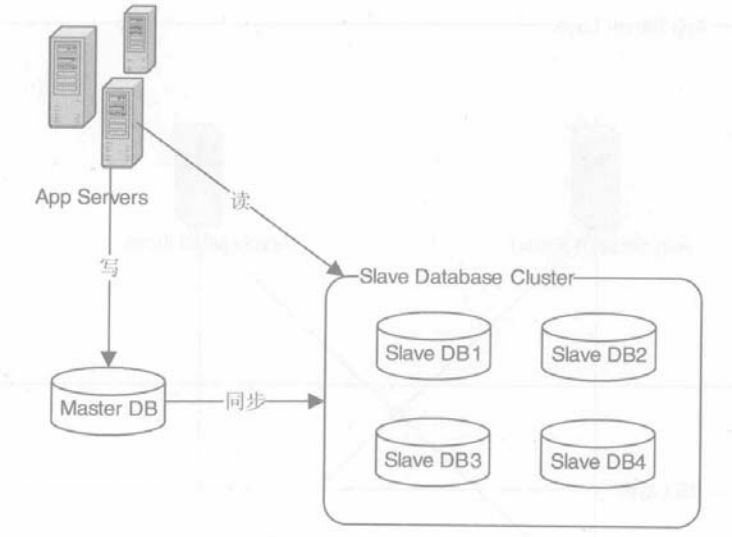


图 6-6　数据库层的读/写分离的部署模型

尽管 Slave Database Server 数量是可以线性扩展的，但是基于以下两个原因，Slave Database Server 也不是越多越好。

（1）如果应用读/写比例不是很悬殊，单纯增加 Slave Database Server 对于 SaaS 应用性能提升并没有特别的作用，通常情况下，Slave Server 的数量与读/写比例对应。例如，读/写比例为 80%读 20%写的 SaaS 应用，Slave Server 的数量以 4 台为宜。

（2）Slave Database Server 过多可能造成 Master-Slave 之间的同步性能降低。

总之，读/写分离技术是提升数据库层伸缩性的一个重要手段。但是，当应用的性能瓶颈出现在写操作之上的时候，读/写分离技术对于伸缩性的提升就非常有限。我们还需要进一步提升数据库层伸缩性的方案。

6.3.3　数据库的水平切分

无论是数据库的垂直切分，还是读/写分离，对于实现数据库层的水平扩展，适用范围都比较狭窄。我们还得继续寻找更普遍适用的数据库层的水平扩展方案。

我们可以先回到第 3 章来看看，在实现 Multi-Tenant 的三种方案中，我们为什么必

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

须要求所有用户共享一个 DB，共享一个 Schema 呢？虽然，在第 3 章中，我们最终选择了这个共享 DB、共享 Schema 的方案，而放弃独立 DB 以及共享 DB、独立 Schema 的方案，主要是因为无论采用独立 DB 还是采用共享 DB 独立 Schema，在非常大的用户量的情况下，每个租户一个 Schema 的方案运行维护成本将会很高。这样，很难提升 SaaS 的规模效应。我们并没有限定所有租户必须共享一个 DB 和共享一个 Schema。也可以每 10 000 个租户共享一个 DB，再增加 10 000 个用户则增加另一个共享 DB，这可能是更为合适的策略。因为，这一方面可以更好地满足水平扩展的需求，另一方面数据库的数量也不会太多，运行维护成本不会增长得太快。就算未来几年内达到 100 万个租户，也只需要 100 个 DB。

那按照租户划分 DB 的方案在业务上可行吗？对于大部分 SaaS 应用，答案是肯定的。原因是，SaaS 应用的不同租户之间在业务上没有任何关联，租户之间的数据是完全隔离的。只有很少部分的共享数据（整个系统全局性的一些配置信息），但它们通常都是只读的，是不允许任何一个租户更改的。

接下来就该进行具体的架构改造了。

（1）首先，租户和用户数据必须位于一个集中式的数据库中。原因是，在用户登录之前，系统是无法预知其属于哪个租户的。

（2）租户对应到哪个物理数据库的映射，通常有两种处理方式：

- 一种是采用 Hash 算法。采用 Hash 算法实现更为简单，性能也更高，但是扩展性略差。因为 Hash 算法是一开始就确定的，如果后面变更的话，会涉及租户数据的迁移。
- 另一种则是将租户对应到哪个物理数据库也作为关系表存储在集中式的租户数据库中。这种方式更为常见。在用户登录时，通过查询相应的关系表，即可确定其对应租户的业务数据存储在具体哪个物理数据库中。

改造后的系统部署模型如图 6-7 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

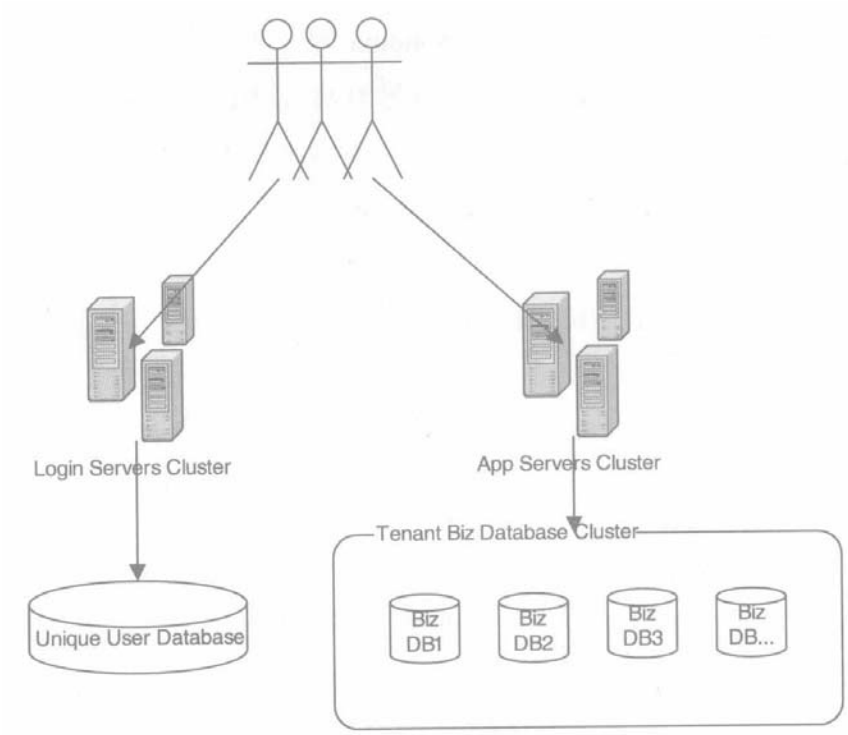


图 6-7　数据库层的水平切分后的系统部署模型

该架构从逻辑上已经实现了数据库的水平切分，不同的租户数据被分布到不同的物理数据库上。随着租户数量的增加，仅仅需要通过增加新的租户业务数据库，并将新的租户映射到新增的租户业务数据库上即可。在大部分情况下，该架构已经比较完美地实现了数据库层的水平扩展。

但是如果应用规模进一步扩大，租户数量在持续增加，应用服务器层和数据库层都在持续地水平扩展。再增加新的服务器的话，该架构还是存在一丝隐患的。由于数据库连接是一种创建成本较高且较为稀缺的资源，而上述架构中的每台应用服务器需要连接到每台数据库服务器上。这样，当应用服务器数量扩展到一定数量时，数据库服务器的连接数将可能成为系统的瓶颈。出现这个问题的原因是，尽管应用服务器层和数据库层已经分别实现了水平扩展，但是由于其彼此之间没有任何对应关系，导致所有的应用服务器要与所有的数据库服务器关联（以 m 台应用服务器和 n 台数据库服务器为例，它们之间的关联有 $m \times n$ 个）。

要解决这个问题，只需对上述架构做出微小的调整。所有的应用服务器不应该是完

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

全平等的，应该与数据库服务器对应。也就是说，不同的租户可能不仅有不同的数据库服务器，也对应有不同的应用服务器，每组应用服务器都仅连接对应的数据库服务器。调整后的部署模型如图 6-8 所示。

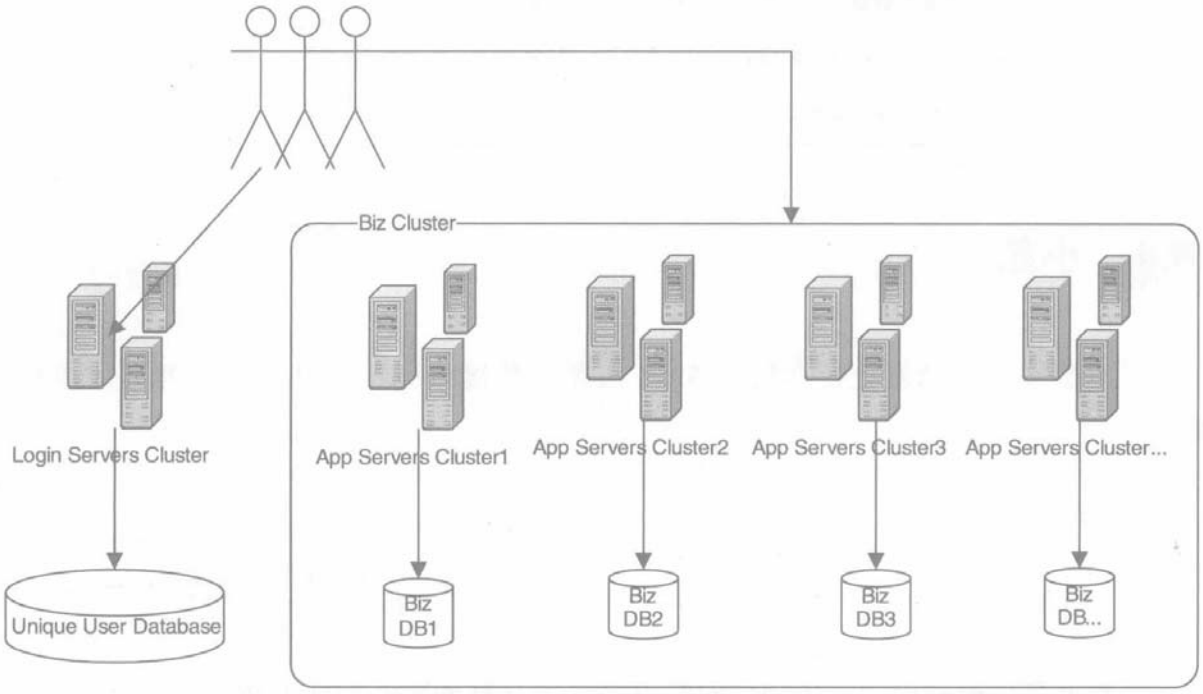


图 6-8　改进的数据库层的水平切分的部署模型

6.3.4　三种数据库层的水平扩展方案对比

应用层的三种水平扩展方式之间有互斥和互相替代的关系。与此不同的是，数据库层的三种水平扩展方式是从不同的角度实现数据库层的水平扩展的，显然，这三种扩展方式其实是互补的，可以共同采用。表 6-2 对这三种数据库层的水平扩展方案进行了比较。

表 6-2　三种数据库层的水平扩展方案比较

实现方式	优　点	不　足
垂直切分	■ 实现简单	■ 扩展能力有限 ■ 强关联的应用不容易垂直切分

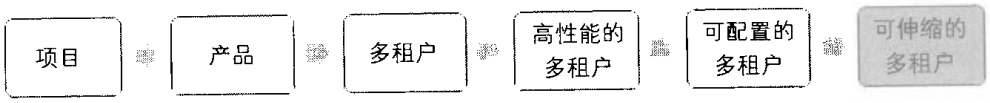
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

续表

实现方式	优　点	不　足
读/写分离	■ 可有效分担读的压力 ■ 主要在数据库层扩展，应用修改较小	■ 对于读的比例不高的应用，扩展能力有限 ■ 依赖于数据库本身的同步能力
水平切分	■ SaaS 应用中普遍适用 ■ 扩展性强，基本支持无限扩展	■ 实现比较复杂，应用一般需要做较大改造 ■ 需要预先做好负载规划，后期数据迁移比较困难

6.4 小结

不经意间，郭靖就已经实现了 SaaS 应用成熟度级别的最高级：可伸缩的多租户架构。



本章中我们分别描述了应用层和数据库层的几种伸缩性实现方案：

- 应用层的水平扩展方式
 - ✧ Session 复制；
 - ✧ Session Sticky；
 - ✧ 基于 Cache 实现集中式 Session。
- 数据库层的水平扩展方式
 - ✧ 垂直切分；
 - ✧ 读/写分离；
 - ✧ 水平切分。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

应该说，应用层和数据库层的水平扩展只是应用可伸缩性的一部分，上面提到的几种实现伸缩性的方案，也仅仅是 SaaS 应用中最常用的方案。对于更大型、更复杂的应用，可能需要更为通用的分布式解决方案。这些方案不仅仅适用于 SaaS 应用，而且普遍适用于大型互联网应用。我们将在第 3 部分“SaaS 基础架构”中对分布式技术做更为详细的介绍。

另外，可能还有读者会问，杨康找来的那个 SaaS 软件平台是什么呀？怎么会有这么大的市场潜力，一下子就能带来数以万计的客户？呵呵，这里我们就暂且卖个关子。到最后 1 章“开放的 SaaS 平台”中大家就会了解到这个平台的魔力。

SaaS 架构设计

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 7 章

SaaS 系统安全

7.1 应用安全 ……150
7.1.1 身份认证 ……150
7.1.2 权限管理 ……153
7.1.3 日志记录 ……156
7.1.4 应用监控 ……157
7.2 数据安全 ……158
7.2.1 数据隔离 ……158
7.2.2 数据库连接安全 ……159
7.2.3 敏感数据加密 ……160
7.2.4 数据量监控 ……164
7.3 网络安全 ……165
7.3.1 安全传输 ……165
7.3.2 网络攻击防范 ……167
7.3.3 网络监控 ……168
7.4 小结 ……168

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

一日，杨康正与郭靖在办公室总结最近两个月公司的运行情况。

郭靖：最近的销售进展好像不是很理想？

杨康：现在的销售很难开展了，熟悉的能拉上关系的都已经是我们的客户了。

郭靖：那我们要开拓渠道，向全国乃至全世界的客户销售。

杨康：我也是这样想啊，但人家不相信我们的系统啊！

郭靖：主要是什么原因呢？

杨康：毕竟与传统应用不一样啊，整个系统都在我们这里，客户觉得不安全，觉得不可靠了。

郭靖：嗯？

杨康：因为我们的系统上，有那么多的租户在使用，客户很害怕别的客户登录上去以后，也能看到他的数据，或者操作他的数据。

郭靖：这应该是不会的啊，我们做了身份认证和权限校验的，我们的系统是做了数据隔离的啊，别的客户是没有权限看他的数据的。

杨康：还有就是客户觉得他的数据是很重要的，比如，重要客户的联系方式，上次不就有个客户的数据被删除了。

郭靖：那个肯定是他们公司的员工误操作，不小心把数据删除了。

杨康：可是我们没有地方可以查询用户的操作记录，没法说服客户的。那次还是小事，如果真的所有客户资料、联系方式等被泄露，我们的麻烦可就大了。

郭靖：是啊，我们得想法把客户联系方式这种重要资料进行加密存储。看样子，我们现在得把系统里的安全问题都清理一遍，在没有出现问题前，把它们解决掉。

杨康：嗯，只有系统安全了，用户才敢放心使用，我们的市场才好做啊！

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

郭靖一边说着，一边陷入了深思：这个 CRM 系统哪些地方还有安全问题呢？怎么解决呢？

从上面的对话中可以看出，很多客户在使用 SaaS 应用时，有很多的顾虑和担忧；同时，SaaS 应用系统也会受到众多的安全考验。

在传统应用中，应用服务器和数据库设备都是由客户自己掌握，每个客户的数据自然是完全独立互不干扰的，这可以让客户觉得很安全、很踏实。而在 SaaS 应用中，服务器不再由客户自己管理，客户开始觉得不放心了。数据存到哪里了？会不会丢失？另外，传统应用总是由一个客户独占使用，而 SaaS 应用是有很多租户同时在使用的。租户之间的使用，会不会出现冲突？数据会不会被其他租户看到？以上这些都成了客户使用 SaaS 应用的顾虑。

另外，SaaS 应用是完全基于互联网使用的，所有的交互和数据都需要通过互联网。这与传统的局域网应用或者单机应用相比，会更容易受到黑客的攻击，受到的安全考验会大很多。这就要求 SaaS 应用在设计时，需要更加系统地考虑安全问题，并采用更可靠的安全技术和手段来保证系统的高可用性，保证数据的完整性和保密性。

对于 SaaS 来说，安全确实是一个备受关注的问题。下面从应用安全、数据安全、网络安全三个方面，阐述 SaaS 应用要考虑的安全事项及相关技术。

7.1 应用安全

首先要考虑的是应用安全，要保证客户在使用 SaaS 应用时能享受到与传统应用一样的高可用性和高可靠性。在身份认证、权限管理、日志管理、应用监控等方面采取更多的安全措施。

7.1.1 身份认证

身份认证就是实现对客户身份的识别和验证，这是保证整个系统应用安全的基础。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

通过严格的身份认证，防止非法用户使用系统，或伪装成其他用户来使用系统。

实现身份认证，可以采用集中式认证、非集中式认证和混合认证三种方式。

（1）集中式认证

集中式认证，就是由 SaaS 应用系统提供一个统一的用户身份认证中心。所有租户都到这个用户中心来管理和维护各租户的身份数据，SaaS 应用直接到统一的认证中心对用户身份进行校验。这种认证方案可以参见图 7-1。

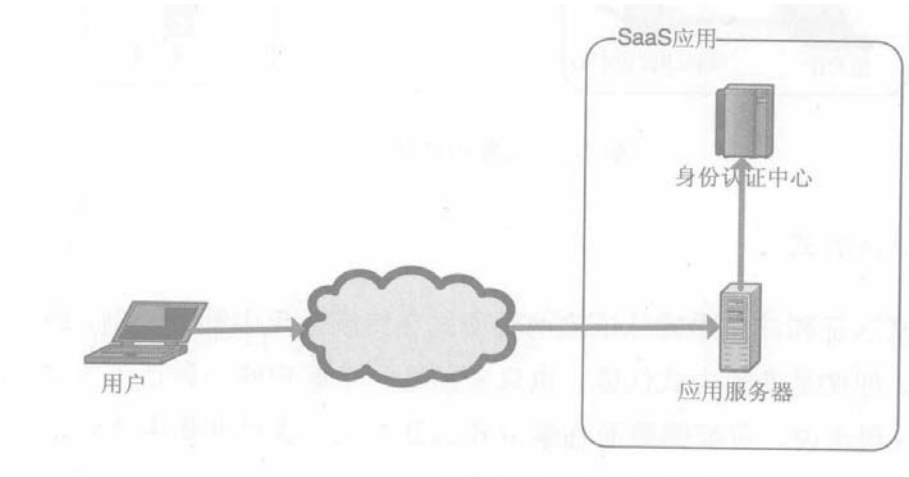


图 7-1　集中式认证

集中式认证在用户身份的安全性上更容易得到保障，但需要租户在 SaaS 应用的身份认证中心维护自己的用户，不能复用已有系统的用户信息，不能与其他系统一起统一维护。并且，难以实现与租户其他系统之间的单点登录。

（2）非集中式认证

非集中式认证，就是各租户使用自己的用户身份认证系统进行身份校验，并且发布安全令牌。SaaS 应用实现一个身份中心，负责接受安全令牌，并据此获取用户身份信息，允许用户使用系统。这种认证方案可以参见图 7-2。

非集中式认证，最大程度地复用租户现有的认证中心，能够实现用户身份的统一维护，容易实现租户各个系统之间的单点登录。但是，这种认证方式的安全性依赖于各租

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

户自身的认证系统。由于各租户在管理上及对安全性的要求是不一致的，这将导致 SaaS 应用身份认证的安全性容易出现短板。

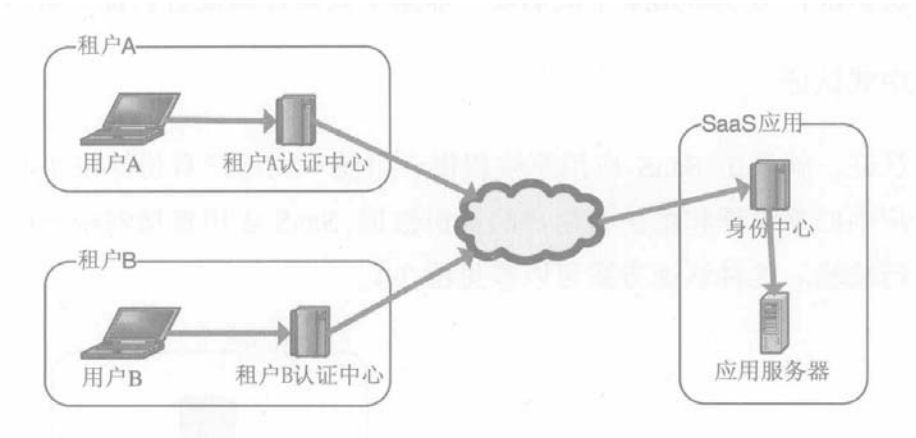


图 7-2　非集中式认证

（3）混合认证方式

尽管集中式认证和非集中式认证这两种方式在传统应用中都会遇到，但总是会只面对其中的一种，即使是非集中式认证，也只会面临一个客户的一种认证方式。而对于多租户的 SaaS 应用来说，可能需要面临集中式认证方式与多种非集中式认证方式同时存在的混合认证方式。这种认证方案可以参见图 7-3。

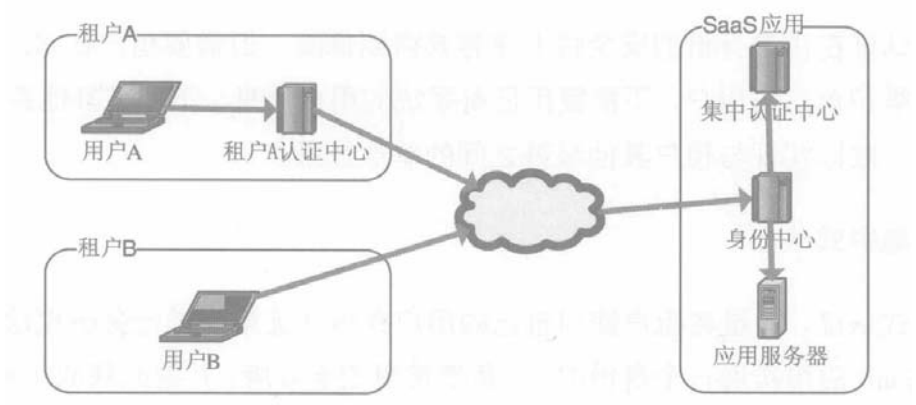


图 7-3　混合认证方式

一般情况下，大多数中小型租户自己并没有专门的身份认证中心，要求 SaaS 应用本身提供集中的认证中心；而对于较大的租户，可能已经建立起自己的身份认证系统。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

这就要求 SaaS 应用既要实现统一的集中认证中心，供大多数中小型租户使用；同时，还需要支持部分租户自身的身份认证系统。

对于 SaaS 应用来说，这种方式实现起来会非常复杂。设计者不仅要考虑前面两种方案中的所有内容，还要考虑两种方案同时运行，以及在租户之间的切换。另外，系统的安全性与非集中式认证一样，依赖各租户自身的认证系统。

（4）方案比较

三种认证方案在安全性、灵活性、用户使用体验、用户维护难度、实现复杂性等方面都存在较大的差别。三者的比较可以参见表 7-1。

表 7-1　身份认证方案比较

比 较 项	集 中 式	非集中式	混合方式
安全性	高	低	低
灵活性	低	中	高
用户使用体验	不好	好	好
用户维护难度	高	低	低
实现复杂性	低	中	高

7.1.2　权限管理

权限管理，就是要实现用户使用系统功能的访问控制，保证有效用户正常使用系统的同时，防止非法用户和无权限用户对系统功能的使用。与传统应用的权限管理不同的是，SaaS 应用的权限管理需要对租户的权限进行控制，并且基于租户的授权模式。

下面从权限模型、权限分配模式、权限校验三个方面，对 SaaS 应用中的权限管理与传统应用的一些差别进行介绍。

（1）权限模型

对于权限模型，传统系统广泛地采用 RBAC（Role-Based policies Access Control）

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

模型体系。RBAC 模型由美国国家标准与技术研究院（The National Institute of Standards and Technology，缩写为 NIST）制定。标准 RBAC 模型包括 4 种，分别是基本模型 RBAC0（Core RBAC）、角色分级模型 RBAC1（Hierarchal RBAC）、角色限制模型 RBAC2（Constraint RBAC）和统一模型 RBAC3（Combines RBAC）。

RBAC0 模型基本原理见图 7-4。

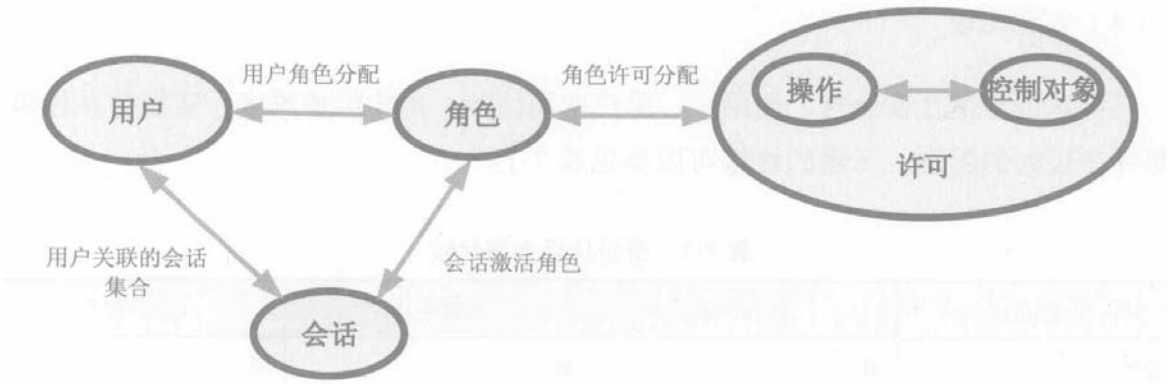


图 7-4　RBAC0 模型图

在多租户的 SaaS 应用中，RBAC 模型体系依然适用。但是，模型中涉及的用户、角色、操作及操作对象，在传统应用中都是属于系统全局的，而在 SaaS 应用中，情况就变得比较复杂了。其中，用户都是按租户隔离的；而角色既会有全局的，也会有租户自身的，而租户的角色是需要进行隔离的。从系统功能层面讲，操作及控制对象虽然是全局的，但各个租户所购买的功能范围也是不一样的，这样的话，生成的许可租户之间也是不一致的。

（2）权限分配

从上面的权限模型可以看出，权限分配主要包括三个方面：角色管理、用户到角色的分配、角色许可的分配。

角色在权限管理中发挥着很重要的作用。对于传统应用来说，所有的角色都是全局的；而在 SaaS 应用中，既会有全局的角色，也会有租户相关的角色。全局角色可能又包括系统管理员和各租户都会通用的角色。系统管理员主要负责对系统的配置和租户的

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

管理；系统通用角色主要是为简化租户的操作，统一定义的系统操作角色。租户相关的角色，又可以分为租户管理员和租户自定义的业务角色。租户管理员负责租户自身的一些设置、自定义角色管理及租户的权限分配。SaaS 应用角色的分类可参见图 7-5。

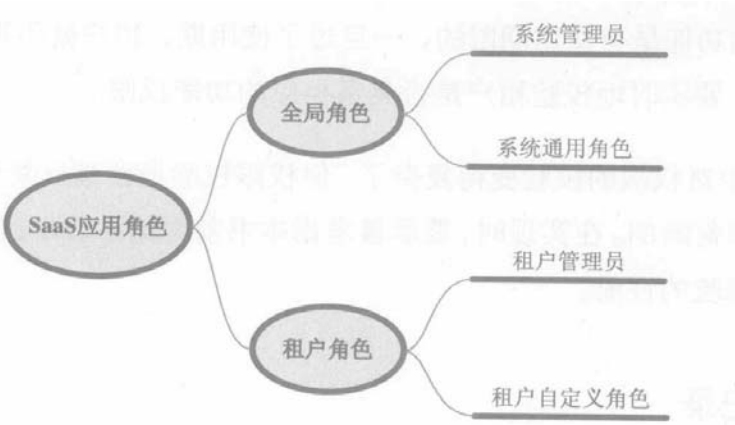


图 7-5　SaaS 应用角色的分类

SaaS 应用中的用户基本上是由租户自身来管理的，而用户到角色的分配也是由租户管理员来完成的。租户可以建立自己的角色，也可以使用系统通用角色。租户可以将自己的用户分配给系统通用角色，也可以分配给自定义的角色。

角色许可的分配，包括两部分：一部分是系统通用角色的许可分配；另一部分是租户自定义角色的许可分配。系统通用角色的许可分配是由系统管理员统一完成的，可以对系统内的所有许可进行分配。而租户自定义角色的许可分配则由租户管理员来完成。但在 SaaS 应用中，如前面有关功能可配置性章节所述，不同的租户所购买的功能是不一致的，那么这个租户的所有用户都只能使用其所购买的功能。这需要在授权的时候只能对租户所购买的功能进行授权。

（3）权限校验

在权限校验时，应该首先校验租户是否已经购买了相应的功能。如果购买了，再来校验该用户是否被授权使用该功能；如果没有购买的话，就不要再校验是否被授权了。之所以要这样做有两个原因：

首先，由于系统内有些角色是系统全局的，这些全局的角色，可能被授予了比较多

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

的功能权限，但是并不是每个租户都购买了这些功能。因此，在权限校验时，应该首先校验租户是否已经购买了相应的功能。

其次，尽管在系统内对用户进行授权时，已经对租户是否购买功能进行了校验。但是，租户购买的功能是有使用期限的，一旦过了使用期，租户就不再具有使用该功能的权限了。所以，要实时地校验租户是否具备相应的功能权限。

SaaS 应用中对权限的校验变得复杂了，但权限校验是会被反复调用的，对权限校验的性能要求是非常高的。在实现时，要尽量考虑本书前面的章节中谈到的性能优化手段，尽量提高权限校验的性能。

7.1.3 日志记录

如果用户在系统中做了一些错误的操作，导致租户的重要数据丢失或者出错了，租户可能会怀疑这些错误是由于 SaaS 系统的原因造成的，或者是其他租户的操作造成的。例如，在 CRM 系统中，一个用户的误操作可能将一些重要的客户资料删除了，这对于租户将是非常重大的损失，那么，如何向租户证明这是由于用户的误操作造成的呢?

日志记录就是要对用户在系统中的操作行为和操作的数据等进行记录，以便对用户在系统中的操作进行查证，以保证用户行为是不可伪造的、不可销毁的、不可否认的。也就是说，用户在系统中的行为是有据可查的，不能在系统中伪造自己的行为，或者伪造其他用户的行为；同时，用户是不能销毁这些证据的，不能否认自己的行为。

日志记录具体包括两部分：行为日志记录和数据日志记录。

（1）行为日志记录

行为日志记录就是要对用户在系统中所访问的每一个页面，在各页面中所做的每一个行为都记录下来，记录用户的身份和行为的时刻。例如，租户 A 的用户 A1 在 2008 年 01 月 01 日 1 时 1 分 0 秒访问了 CRM 系统的重要客户列表页面，做了删除客户信息的操作。行为日志记录的实现，可以采用面向方面的方案来实现，例如，通过过滤器或拦截器的方式，来对所有的页面请求行为及页面里的提交行为都进行拦截，然后将其记

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

录在日志文件里或数据库里。行为日志记录是辨别用户在系统中行为的一个重要依据，对于系统使用与系统运营分开的 SaaS 系统就显得尤为重要。

（2）数据日志记录

数据日志记录，就是要对用户在系统中所操作的数据进行记录，记录数据的变更过程及变更的历史。这在多人操作同一个数据的系统中显得尤为重要。例如，用户 A 在财务系统中提交了一张财务报销单，报销金额是 1000 元，在经过了用户 B、C、D 等一系列人的修改和审批以后，用户 A 看到的报销金额变成了 500 元，如果没有报销金额的变更日志记录，用户 A 一定会很疑惑，是谁因为什么原因修改了这个报销金额。那么，系统就很有必要对报销金额的变更进行日志记录。

（3）日志记录的安全

日志记录是对用户在系统中行为进行查证的依据，是用来跟踪和保障系统安全的，那么，日志记录本身的安全性也是需要重点考虑的。首先，日志记录应该是只读的，最好能加上时间戳，不应该被人为修改或者伪造；其次，日志记录涉及用户的隐私，应该是保密的，要防止被非法使用。租户的日志只向 Tenant 管理员开放，并且 Tenant 管理员也只能查询租户自己的日志。

7.1.4 应用监控

对于 SaaS 应用来说，由于大量的客户都在通过互联网使用系统，**要特别防止非法客户或个别客户的恶意攻击，造成其他客户的不能正常使用。**需要对应用进行监控，以保障应用的高可用性和高可靠性。

可用性监控，就是要监控应用的硬件和软件的可使用状况，防止系统出现非正常的停机、拒绝服务等情况。SaaS 应用是完全基于互联网使用的，系统的可用性更容易受到来自网络的各种攻击，系统在受到攻击后可能出现不可用的状况。系统一旦不可用，将给客户带来很大的损失，可用性监控就是要及早地发现系统的不可用状况，并通知管理者。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

可靠性监控，就是在应用可用性监控的基础上，对系统的运行状况是否可靠稳定进行监控。SaaS 应用完全基于互联网进行使用，容易受到黑客的攻击。另外，系统长期处于高负荷运行的情况下，也可能会出现系统不稳定的状况。因此，要对系统运行的状态进行实时监控。

对于应用监控的常见实现方案是采用心跳监测。在应用中部署一个服务，以获取应用是否可用、系统硬件及软件等运行时数据，通过监控中心对这个服务进行“心跳”式的定时访问，以检测系统的可用性和可靠性。如果监控到系统出现不可用或不可靠的情况，可以采用告警机制，给管理者进行短信或邮件的提示。如何判断系统处于不可用或不可靠的状态？需要制定包括 CPU 使用率、内存使用率、服务响应时间等各种参数，并为这些参数设定指标阈值。当检测系统的一些指标超出阈值时，发出告警通知。

7.2 数据安全

对于 SaaS 系统来说，客户最关注的问题就是数据的安全。要保证客户的数据安全，需要做到对租户进行数据隔离，保证数据库访问的安全性，对敏感数据进行加密。

7.2.1 数据隔离

传统的应用因为每个客户的设备是独立的，相互之间数据是绝对隔离的，而且应用也是相互独立的。SaaS 应用相对于传统应用来说，应用部分不再是隔离的，而数据设备可能不再是独立的了，至少对于租户来说是不可预知的。这就让用户觉得他的数据不再那么安全了。如何保证用户数据仍然像传统应用一样安全呢？

数据隔离，就是要对租户之间的数据进行隔离，以确保各租户数据的完整性和保密性。也就是说，租户 A 的数据始终只能被租户 A 的用户查看和使用；租户 A 的用户也只能查看和使用租户 A 的数据。这一点对于 SaaS 应用来说，是至关重要的。

有关数据隔离方案的实现，在前面第 3 章中已经做了详细介绍，在此不再讨论。在进行 SaaS 应用设计时，可以综合考虑业务的安全性级别和用户愿意承担的成本。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

7.2.2 数据库连接安全

在传统应用架构中，为了确保存取数据的安全性，在数据库连接上常采用两种方法：一种是模拟操作，另一种是受信任的子系统账户。而对于 SaaS 应用来说，这些方法仍然是适用的，只是实施起来更加复杂了。

（1）模拟存取

采用模拟存取方法，就是在数据库中设置不同用户对数据库对象具有不同的操作权限，系统直接采用当前用户的身份去模拟操作数据库对象。这些数据库对象包括各数据表格、视图、存储过程等。由数据库根据用户的操作权限，来判断该用户是否可以对相应的数据库对象进行相应的操作。具体方案参见图 7-6。

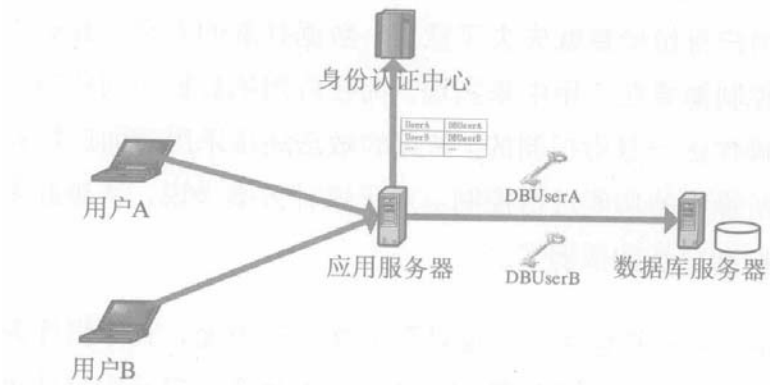


图 7-6 模拟存取

这种方法是非常安全的，数据存取的安全性由数据库来控制，但这种方案实施起来是很困难的。尤其是对于多租户的 SaaS 应用来说，各租户的用户是由租户自身来维护的。但是，不可能让租户直接去设置数据库中各数据库对象的权限。显然，这种方案实施起来是很困难的。

（2）受信任的子系统存取

采用受信任的子系统存取方法，就是不管当前用户的身份如何，应用总是使用一个特定的用户身份与数据库连接，操作相应的数据库对象，而数据库系统总是信任这个用

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

户具有这些数据库对象的相应的操作权限。具体方案参见图 7-7。

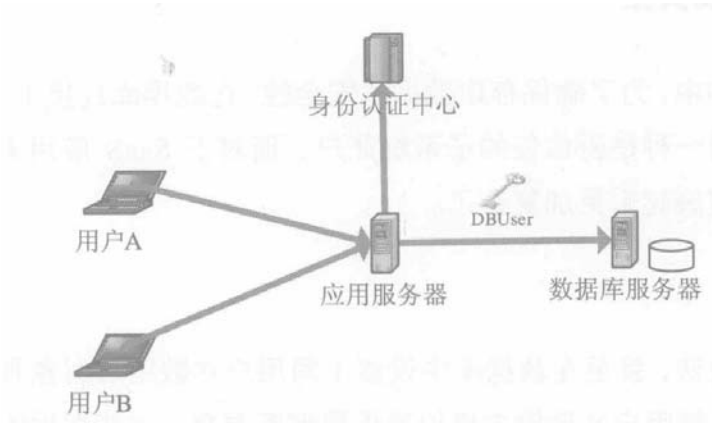


图 7-7　受信任的子系统存取

在这种情况下，所有的系统用户都在使用同样的数据库用户身份来操作数据库对象，数据库系统的用户身份检查就失去了意义，数据存取的安全，数据库系统就无法控制了。相应的安全控制需要在应用中来实现，而在应用的数据访问代码中，一般是很少针对数据库对象的操作进行身份控制的。常见的做法还是采用前面应用安全一节所述的权限管理，对用户所操作的功能进行控制。对于这种方案来说，实施起来是简单了，但数据存取的安全性控制显然就很弱了。

在 SaaS 应用中，可以综合考虑上面两种方案。一方面，在数据库系统中根据要对数据库对象所做的操作，设置不同的数据库用户，而每个用户身份只能进行相应的数据操作，并不具备其他数据库对象操作的权限；另一方面，在系统的应用中对相应的系统功能与对应的数据库用户身份之间进行匹配，然后根据当前用户所操作的系统功能，使用相应的数据库用户身份去连接数据库，并对数据库对象进行相应的操作。这种方案既利用了数据库的用户身份控制，限制了各数据库用户能对数据库对象进行的操作，同时实施起来也比较容易。

7.2.3　敏感数据加密

在传统应用中，对用户身份密码要进行加密是比较常见的，但很少对用户的业务数据进行加密。因为，传统应用的数据库是由用户自己管理的，一般认为数据库管理员是

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

值得信任的，不需要对业务数据进行加密。

然而，SaaS 应用的数据库是由运营商来管理的，而对于租户来说，运营商及数据库管理员并不是完全值得信任的。对于租户的一些敏感数据，可能是不愿意让任何人看到的，更不愿意运营商使用和统计这些数据。例如，公司的财务数据。如果把这种数据以明文的方式保存在数据库中，租户显然是不敢使用这样的 SaaS 应用的。

另外，即使保证了数据库及数据库连接的安全，仍然很难保证租户的数据不会被非法获得，例如，病毒的原因，或者黑客的攻击，或者管理上的疏忽。如果这些数据是很敏感、很重要的，如关键客户的联系方式、公司的财务数据等，那么，这种数据的泄露可能会给租户造成很重大的损失。

针对上面两种情况，可以考虑对系统内的敏感数据进行加密。一旦数据被加密以后，就变成了不可识别的密文。这些数据即使被泄露，也能保证数据的安全性。

（1）加密算法选择

就加密算法而言，可以分为可逆加密和不可逆加密；而可逆加密算法又可以分为对称加密和公开密钥加密，公开密钥加密又称为非对称加密。加密算法分类可参见图 7-8。

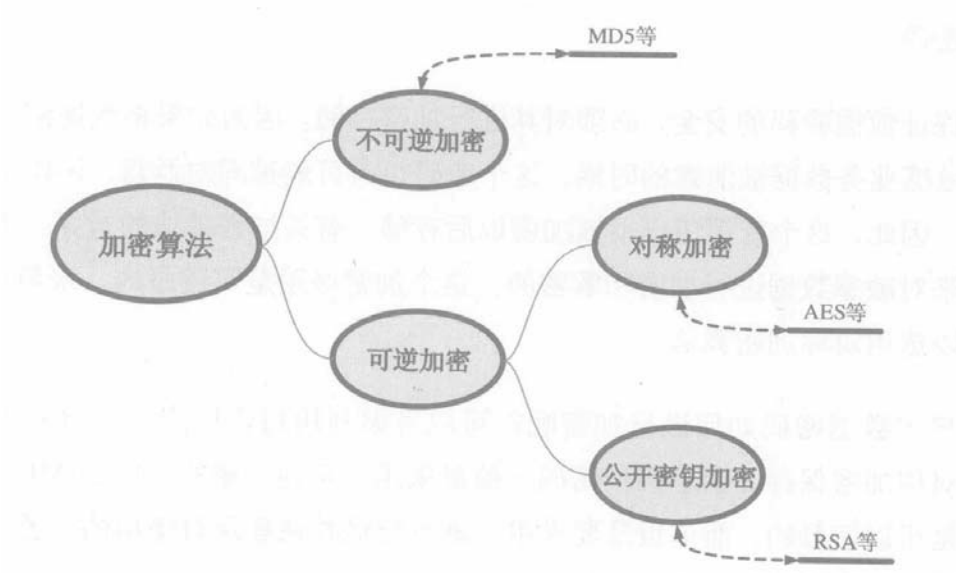


图 7-8　加密算法分类

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

不可逆加密算法如 MD5，数据被加密以后是不可以被解密的，即无法根据密文来得到明文。MD5 算法常用于对校验数据进行加密，对于租户的用户身份密码，是可以采用 MD5 算法进行加密的。由于租户的业务数据是需要还原使用的，对租户的业务数据进行不可逆加密是不合适的。

对于租户的敏感业务数据，应该采用可逆的加密算法。具体应采用对称式加密，还是非对称式加密呢？对称式加密就只需要一个密钥，既可以用这个密钥将明文加密成密文，也可以用它将密文解密成明文。而非对称式加密则需要两个密钥：一个公钥和一个私钥，公钥用来将明文加密成密文，而私钥用来将密文解密成明文。

对于租户敏感业务数据的加密，采用对称加密或公钥加密都是可以的，但二者在加密和解密过程中的计算强度是不一样的。公钥加密会比对称加密的计算时间要长，有将近 2～3 个数量级的差别，从系统的性能考虑，还是建议采用对称式加密。

（2）数据密码的产生与管理

敏感业务数据加密是针对租户而言的，而一个租户可能有很多用户同时在使用数据。在同一个租户的多个用户之间，要考虑数据共享的问题。不能因为 A 用户保存数据时对数据进行了加密，而导致 B 用户不能使用该数据。所以，一个租户应该设置一个统一的数据密码。

为了保证数据密码的安全，必须对其进行加密存储。因为如果将数据密码以明文记录的话，敏感业务数据被泄露的时候，这个密码也有可能被同时泄露，还是不能保证数据的安全。因此，这个密钥还必须被加密以后存储。有关加密算法的选择：因为这个密码是要用来对敏感数据进行加密和解密的，这个加密必须是可还原的，采用可逆加密。另外，可以选用对称加密算法。

对租户的数据密码如何进行加密呢？可以考虑利用每个用户的身份密码来对数据密码进行对称加密保存。由于用户密码一般是采用不可逆加密的，例如 MD5 算法，一般理解为是可以信赖的，而这也是要求用户进行记忆并在登录时使用的。因此，采用用户身份密码对数据密码进行对称加密。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

为了简化用户的使用，同时也提高密码的保密级别，这个数据密码对于租户来说，最好是透明的。可以在创建过程中，由系统自动生成。租户的数据密码 KEY 产生以后，与租户管理员的用户身份数据一起存储。但数据密码是所有租户的用户都需要使用的，所以，在创建租户的其他用户时，需要将其传递给其他用户。也就是说，每一个用户都需要用其身份密码原文对数据密码进行 AES 加密。为了防止数据密码泄露，再将加密过的数据密码密文与用户身份数据一起存储。

数据密码对于用户来说是透明的，而且是租户的所有用户共用的，所以是不能修改的，也没有修改的必要。但是，用户身份密码是可以修改的，数据密码是用户身份密码加密的。如果用户身份密码修改了，数据密码就需要重新用新的用户密码进行加密，再保存。

有关数据密码的产生过程、安全传递，以及用户密码修改时的处理，涉及一些安全上较复杂的处理，在此不做进一步的详细讨论。

（3）敏感业务数据的使用过程

敏感业务数据的使用过程包括对业务数据的加密和解密两部分，使用过程可参见图 7-9。

用户在登录系统时，会输入其用户身份密码进行校验。系统对其输入的用户密码进行 MD5 加密，对加密的用户密码密文进行校验；校验成功后，用其输入的用户密码原文对用户身份数据中的数据密码密文进行 AES 解密，得到租户的数据密码原文 KEY。这个 KEY 始终由身份认证子系统负责管理，并不向外暴露。

系统在使用敏感业务数据时，先从数据库读取数据，读到的是加密数据。调用身份认证子系统提供的接口，对敏感数据进行解密，得到原文，再来使用数据。相反，用户在修改敏感数据并提交时，系统先处理业务逻辑，然后，调用身份认证子系统提供的接口，对敏感数据进行加密，得到敏感数据的密文，再将数据存储。

（4）注意事项

数据加密在一定程度上保证了数据的安全，会给系统带来性能上的损失，在数据存

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

储上，不能在加密后数据列上建立索引，也不能进行模糊查找。因此，建议只对最重要的敏感业务数据才采用此方案。

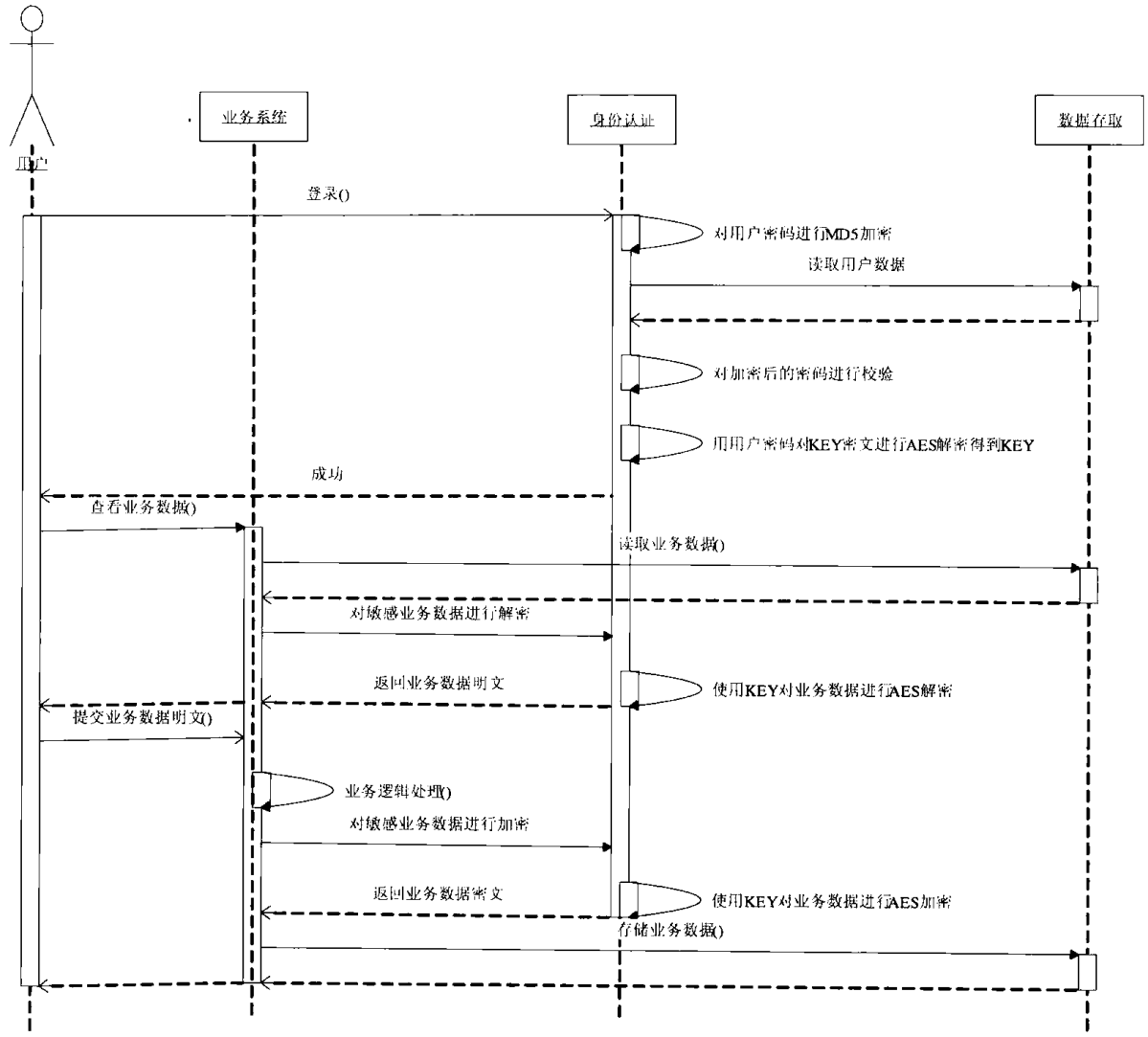


图 7-9　敏感业务数据的使用过程

7.2.4　数据量监控

对于传统应用来说，系统资源是客户独享的；而在 SaaS 应用中，系统资源是所有租户共享的。如果一个租户占用了大量的资源，势必影响其他租户对资源的使用。为了

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

保障所有租户的权利，有必要限制各租户可以使用资源的最大数量。这就需要对各租户使用的资源数量进行监控，包括文件空间的监控和数据库使用量的监控。

文件空间的监控，就是要监控租户在对文件进行操作时，占用系统硬盘空间的数量。由于系统的硬盘空间是有限资源，不可能允许租户无限地使用。而这些空间是由众多租户共享使用的，为了保证所有租户的权利，必须严格控制每个租户对文件空间的使用量。这在许多 SaaS 系统中都很常见，例如，邮件系统都会给每个租户分配最大可使用的硬盘空间。当租户的使用量试图超出了其分配量时，系统则给予其非法操作的提示。

数据使用量的监控，就是要监控租户对系统数据库空间的使用量。与硬盘空间一样，数据库也不是可以无限扩充的。使用数据量的增长也会影响数据访问的性能，如果有租户在系统中不停地新建大量的数据，势必会对系统的稳定运行和可靠性造成影响。因此，要对租户所使用的数据量进行监控。

7.3 网络安全

与传统应用相比，SaaS 应用是完全基于互联网使用的。应用数据在传输过程中更容易受到网络攻击，应用的稳定性也更容易受到网络的影响。SaaS 应用需要高度重视数据的安全传输，对网络攻击的防范，以及对网络进行监控。

7.3.1 安全传输

SaaS 应用是完全基于互联网进行使用的，如果采用明文方式传输的话，将会很容易受到各种各样的网络攻击。这将导致数据的保密性和完整性难以得到保障，应用的安全性也很难实现。因此，对于 SaaS 应用中敏感数据的传输，建议采用安全超文本传输协议 HTTPS 进行传输。与 HTTPS 相关知识请参见下文：

> HTTPS（Hypertext Transfer Protocol over Secure Socket Layer），是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版。即 HTTP 下加入 SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容请看 SSL。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

它是一个 URI scheme(抽象标识符体系),句法类同 http:体系,用于安全的 HTTP 数据传输。https:URL 表明它使用了 HTTP,但 HTTPS 存在不同于 HTTP 的默认端口及一个加密/身份验证层(在 HTTP 与 TCP 之间)。这个系统的最初研发由网景公司进行,提供了身份验证与加密通信方法,现在它被广泛用于万维网上安全敏感的通信,例如交易支付方面。

它是由 Netscape 开发并内置于其浏览器中的,用于对数据进行压缩和解压操作,并返回网络上传送回的结果。HTTPS 实际上应用了 Netscape 的安全套接字层(SSL)作为 HTTP 应用层的子层。(HTTPS 使用端口 443,而不是像 HTTP 那样使用端口 80 来和 TCP/IP 进行通信。)SSL 使用 40 位关键字作为 RC4 流加密算法,这对于商业信息的加密是合适的。HTTPS 和 SSL 支持使用 X.509 数字认证,如果需要的话用户可以确认发送者是谁。

也就是说,它的主要作用可以分为两种:一种是建立一个信息安全通道,来保证数据传输的安全;另一种就是确认网站的真实性。

HTTPS 通信原理请参见图 7-10。其主要的思想包括以下几点:

- 由客户端产生对称密钥 KEY;
- 客户端使用服务器端提供的公钥对 KEY 进行不对称加密后,发送给服务器;
- 服务器端使用自己的私钥进行解密得到 KEY;
- 服务器端和客户端在发送数据时,都采用 KEY 作为密钥对数据进行对称加密;
- 服务器端和客户端在接收数据时,都采用 KEY 作为密钥对接收到的数据进行解密。

HTTPS 对数据安全性的保障,依赖于浏览器的正确实现,以及服务器软件、实际加密算法的支持。另外,HTTPS 不像 HTTP 那样是免费的,是需要到指定的服务器验证,而这种验证服务往往是收费的,所以并不要求所有 SaaS 应用都采用 HTTPS,只是涉及敏感数据传输时才采用 HTTPS,普通的 Web 页面就可以直接采用 HTTP 协议传输。

互联网时代的软件革命
Java学习群:72030155
好资料应该和你的朋友分享,把这本电子书分享给学Java的朋友,Java群空间;可以联系群主获取更多大型企业内部技术教程。

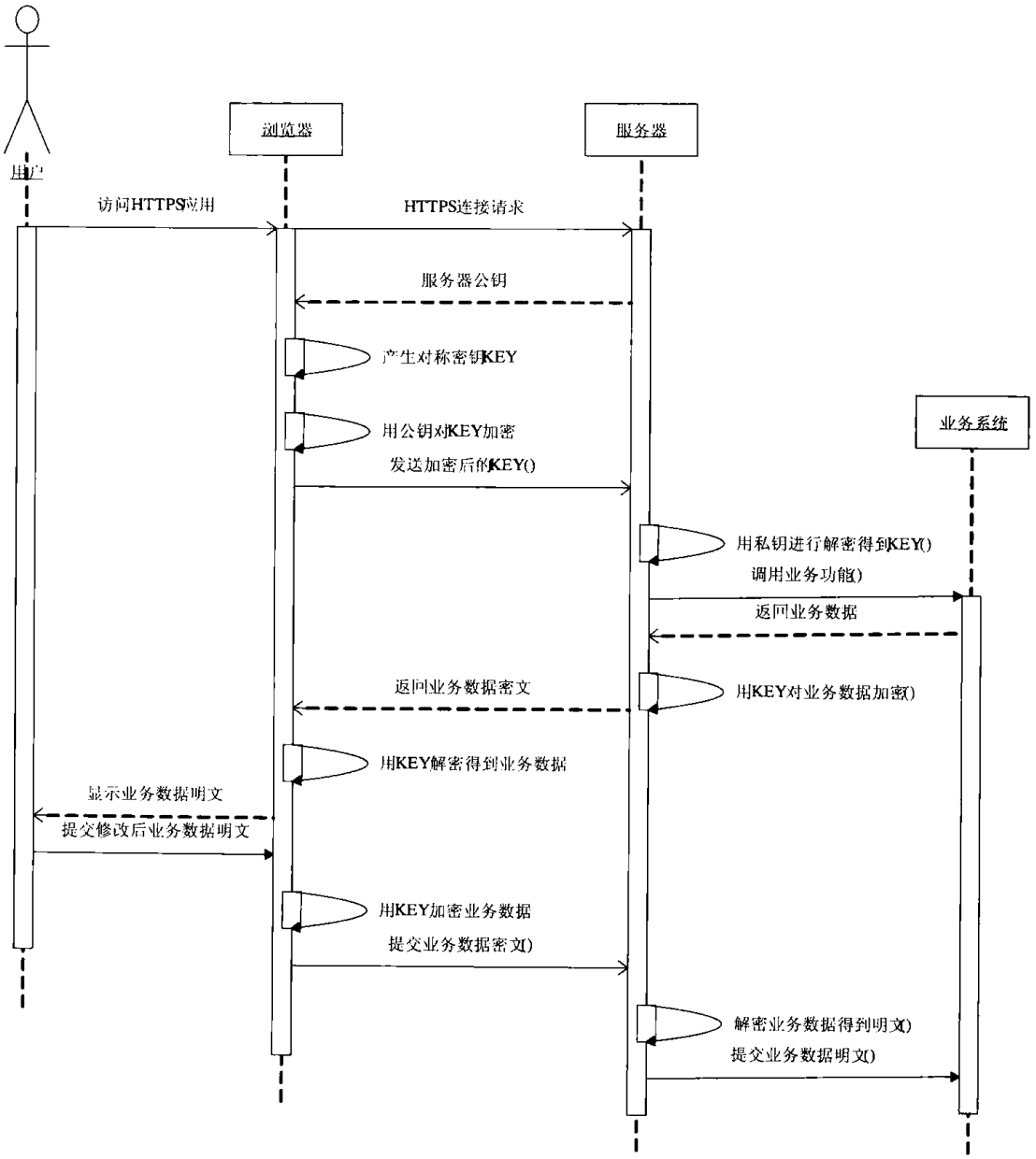


图 7-10　HTTPS 通信原理图

7.3.2　网络攻击防范

虽然 SaaS 应用是完全基于互联网进行使用的，但是如果把整个系统都直接暴露在互联网上，系统将会受到大量来自互联网的攻击。这种网络攻击将会严重影响 SaaS 系

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

统的正常运行和客户的使用。

由于最终用户总是通过互联网来使用 SaaS 应用的，即 SaaS 应用总是需要将一部分内容部署在公网上。但为了保证整个系统的安全，暴露在公网的这部分内容越少越好。因此，要对整个系统进行分层设计和部署，只把系统界面层的部分部署在公网上，而将更重要的应用服务器和数据库服务器部署在防火墙内。具体部署结构可参见图 7-11。

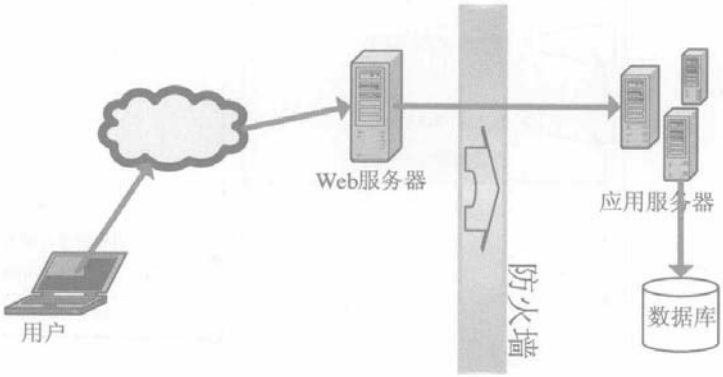


图 7-11　分层部署图

7.3.3　网络监控

网络是否可用，是否稳定，直接影响 SaaS 系统的可用性。除了数据的安全传输以外，SaaS 应用还需要关注网络的可用性、稳定性和传输速度等各个方面。

现在已经有非常成熟的网络监控系统，能够实现对网络设备的运行状况进行 7×24 小时的实时监控。并且，能够在网络出现故障的第一时间进行报警。在此不做过多的介绍。

7.4　小结

在 SaaS 系统使用和推广过程中，系统的安全性受到用户的普遍关注。本章主要从应用、数据、网络三个方面，阐述了 SaaS 应该重点考虑的一些安全事项。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

对于 SaaS 系统的应用来说，需要重点构建系统的身份认证、权限管理、日志记录，确保用户身份的真实性和控制用户操作的权限，保证用户操作的不可否认性，应用安全是整个 SaaS 系统的最重要的安全控制。

对于 SaaS 系统的数据来说，数据隔离与数据库连接的安全都是必须要做到的，这是多租户模式中数据安全的基础。还着重讨论了对于租户的敏感业务数据进行加密的方案，这是让用户可以放心大胆使用 SaaS 系统的一个重要手段。

网络安全是传统应用中不太关注的问题，对于完全基于互联网的 SaaS 系统来说，就显得很重要了，要重点考虑用户数据在网络中的安全传输，保证数据的完整性和保密性。

构建安全可靠的 SaaS 应用，让客户可以安全、稳定地使用系统，是赢得市场的关键和基础。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第8章 离线应用

8.1 系统分析 ……171
8.1.1 离线应用的 4 大核心问题 ……172
8.1.2 离线应用架构 ……173
8.1.3 相关技术背景 ……175
8.2 本地使用 ……176
8.2.1 网络状态检测 ……176
8.2.2 离线页面支持 ……178
8.3 本地存储 ……178
8.3.1 本地数据库支持 ……178
8.3.2 本地数据存储 ……179
8.4 数据同步 ……180
8.4.1 增量数据处理 ……180
8.4.2 多用户冲突 ……183
8.5 数据传输 ……184
8.6 离线应用开发工具介绍 ……186
8.6.1 本地数据库 ……187
8.6.2 本地服务 ……188
8.6.3 工作者线程池 ……190
8.6.4 Gears 待发展的内容 ……192
8.7 小结 ……192

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

这几天，杨康办公桌上的烟灰缸塞满了烟头。他总是皱着眉头、叼着烟在办公室走来走去。看样子是碰到了棘手的事。

“咚！咚！咚！”这时有人敲办公室门。

杨康：请进。

进来一个人，正是公司 CTO 郭靖。

杨康：郭老弟，你看这个该怎么办？

杨康一边说一边把客服部的《用户反馈报告》递给了郭靖。郭靖看到报告上有个大红圈，正是杨康的手迹。

杨康：客户反馈出差途中遇到的烦恼……

神雕科技有限公司：

我购买了贵公司的产品——侠客录 CRM。我用了将近一个月，用得还是很不错的。可我是个经常出差的人，一周至少两天在飞机上，我非常期望可以利用等飞机、坐飞机这段时间处理公司繁忙的业务，也就是说，可以在不上网的情况下使用贵公司的软件。另外最近公司的网络不太稳定，结果无法保存，辛辛苦苦做的报表数据也丢失了。希望贵公司能解决我的烦恼。

一个经常出差的客户：周伯通

8.1 系统分析

郭靖从杨康办公室出来后，召集了手下几个技术骨干开始研究用户的需求。

郭靖：这是一个商务客户非常典型的需求，我先将需求总结一下。

郭靖在白板上以他惯有的速度奋笔疾书。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（1）网络不稳定，如何避免数据保存时不丢失。

（2）离线状态下可以继续使用。

黄蓉：靖哥哥，我感觉这两个需求应该是同一个需求。

郭靖：哦！你说说看。

黄蓉：如果我们解决了离线状态下使用的难题，再出现网络不稳定，系统就会自动切换到离线状态，数据丢失的问题也就解决了。

郭靖：你的意思我理解了，其实就是一个离线使用的需求。

黄蓉：对。

经研究小组分析，客户目前需要的是一个能够支持离线使用的应用，简称**离线应用**。

8.1.1 离线应用的 4 大核心问题

既然是离线应用，则需要在用户机器上存放数据，也就需要系统能够支持本地存储机制。如果应用是 B/S 结构，由于界面是网页，则需要将页面存放到本地，以便离线时启动系统。

由于是离线使用，将会出现服务器端数据与本地数据的差异，数据需要通过同步的方式保持一致。还由于存在多个客户端，它们在与服务端同步数据的时候有可能产生冲突。

同时，从用户体验考虑，还需要解决如何减少网络流量以及数据安全等问题。

通过刚才的分析，离线应用需要解决如下 4 个主要问题：

- 本地使用；
- 本地存储；
- 数据同步及冲突解决；

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 数据传输。

8.1.2 离线应用架构

离线应用架构图如图 8-1 所示。

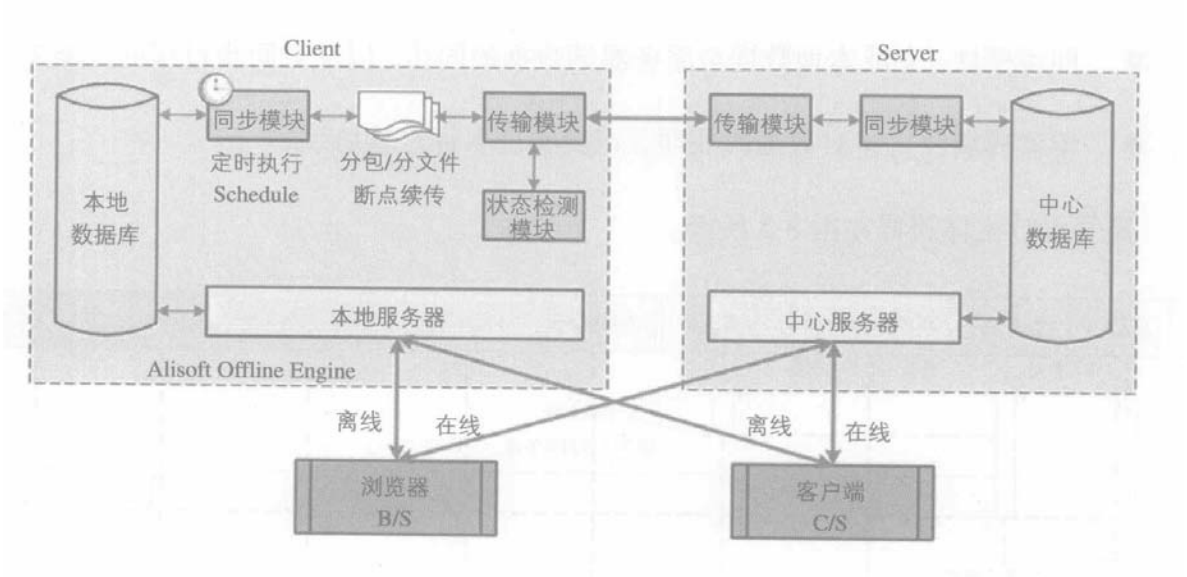


图 8-1 离线应用架构图

整个系统划分为两部分：

（1）本地离线引擎子系统（Alisoft Offline Engine）

- 本地数据库：使用小型文本数据库，可以采用 SQlite 数据库，用于存放业务数据。
- 本地服务器：对于 B/S 应用支持离线状态下访问页面和数据存储；对于 C/S 应用支持数据存储。
- 同步模块：完成本地数据与服务器端数据的同步以及对同步过程的策略支持。
- 传输模块：支持对数据的处理，以及实现各种传输协议。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 状态检测模块：判断当前机器是否在线。

（2）服务器端子系统

- 服务器数据层：与应用服务器交互，从服务器获取数据并把数据提交给“数据同步模块”。
- 同步模块：完成本地数据与服务器端数据的同步，以及对同步过程的策略支持。
- 传输模块：支持对数据的处理，以及实现各种传输协议。

离线设计主体流程如图 8-2 所示。

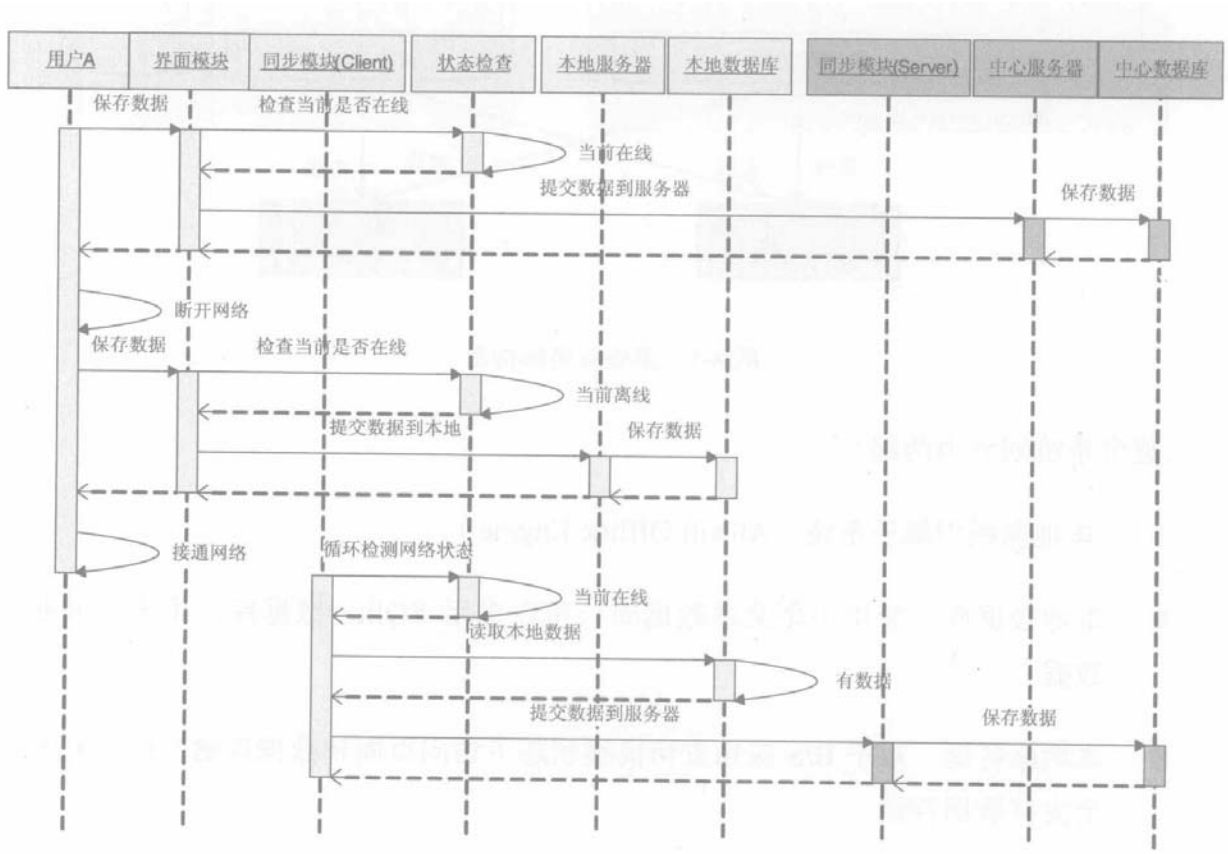


图 8-2　离线设计主体流程图

- 网络连通：网络连接正常时，Web 或 Client 应用直接与 Internet 上的应用服务器通信。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 网络断开：如果网络连接失败，用户依然能够访问，数据接口组件则不向 Internet 上的应用服务器提交数据（即使提交也会失败），而向本地的**本地离线引擎**提交数据，将数据暂存在本地的轻便型数据库里。
- 网络恢复：等网络连接恢复时，再通过“数据同步”模块向 Internet 服务器提交那些待提交的数据。

8.1.3 相关技术背景

在网络应用的开发者群体中，目前正火热地议论着“离线应用”这个概念，以及如何使用离线应用技术。“离线应用”是 Google（谷歌）推出的一种实用技术，并且 Google 将向开发者提供谷歌离线应用开发工具（Google Gears），这是一种为生成离线网络应用软件而开发的开源技术。

这项新产品目前仍处在实验阶段，免费向广大开发者群体提供。谷歌希望能在广大开发者的帮助下测试完善功能和发现产品的局限性。

谷歌离线应用开发工具（Google Gears）的诞生标志着网络应用软件的发展迈出了关键性的一步，因为它解决了用户关注的一个主要问题：在没有网络连接、网速过慢或连接不可靠的前提下，如何离线阅读或浏览网上的一些信息。开发者和用户一样，都希望能够在 Web 和浏览器上完成更多任务——无论是查看电子邮件，进行客户关系管理还是编辑相片，因此增强浏览器环境自身的能力变得愈发重要。

谷歌公司首席执行官埃里克·施密特表示：“借助谷歌离线应用开发工具（Google Gears），我们即将攻克浏览器的一个关键性瓶颈，打造一个更强大的平台，并且让用户获得云中漫步的美妙用户体验。我们坚信，开发者群体有能力将这项新技术发挥到极致，并最终创造出让所有人都可以受益的开放式标准。”

目前 Google Gears 是一个离线应用的开发工具，它并没有解决系统级的数据**同步**时的数据**冲突**问题。本章将阐述有关数据同步问题的分析，详见 8.4 节内容。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

8.2 本地使用

8.2.1 网络状态检测

离线应用，顾名思义，就是能够判断网络状态，客户端根据网络状态决定采用本地或服务器策略。我们可以使用 AjAX 技术的 XML Http Request 来检查网络状态，也可以在 ActiveX 控件中添加对网络状态的检查模块，让脚本可以简单地调用。

```
<script type="text/javascript" language="javascript">
<!--

function isOnline()
{
    var xmlhttp = createHttpObject();
    var url = "http://www.alisoft.com/portal/about/Company.html";
    if (xmlhttp)
    {
        try
        {
            xmlhttp.open("HEAD",url,false);
            xmlhttp.send();
        }
        catch (e)
        {
            return false;
        }

    return xmlhttp.status==200;
    }

    return false;
}

function createHttpObject()
```


互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
{
    var xmlhttp;

    try
    {
        xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
    }
    catch (e)
    {
        try
        {
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch (e)
        {
            xmlhttp = false;
        }
    }

    if (!xmlhttp && typeof XMLHttpRequest!='undefined')
    {
        try
        {
            xmlhttp = new XMLHttpRequest();
        }
        catch (e)
        {
            xmlhttp=false;
        }
    }

    if (!xmlhttp && window.createRequest)
    {
        try
        {
            xmlhttp = window.createRequest();
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        }
        catch (e)
        {
            xmlhttp=false;
        }
    }

    return xmlhttp;
}

-->
</script>
```

使用类似这段 JavaScript 代码中的 isOnline()函数，即可判断网络连接状态，即处于在线还是离线状态。

8.2.2 离线页面支持

为了让 B/S 软件能在离线状态下使用，则必须将页面保存在本地，并且要由本地 Web 服务器来支持。这里的“本地 Web 服务器”和如何保存页面将是问题的关键。

要有本地 Web 服务器的功能，可以通过 IE 插件来完成（如何开发 IE 插件此处不详细介绍，如有兴趣可上网搜索相关资料）。对于用户来说，他照样可以输入应用的 Url 来启动，这些对用户都是透明的，他并不需要关心当前是否在线。该功能属于**本地离线引擎提供**的功能。

8.3 本地存储

8.3.1 本地数据库支持

离线应用还需要解决本地存储，说到数据存储，自然会想到数据库。而这里的数据库需要在用户机器上运行，而 IE 自身没有提供这样的能力，但 IE 可以使用脚本调用

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

ActiveX 控件，因此可以通过 ActiveX 控件集成本地小型数据库来实现数据库的支持。本地小型数据库引擎可以使用 SQLite、Access 这样的文本数据库。

■ 维基百科

SQLite 是遵守 ACID 的关系型数据库管理系统，它包含在一个相对小的 C 库中。它是 D.RichardHipp 建立的公有领域项目。

不像常见的客户/服务器范例，SQLite 引擎不是一个程序与之通信的独立进程，而是连接到程序中成为它的一个主要部分。所以主要的通信协议是在程序语言内的直接 API 调用，这在消耗总量、延迟时间和整体简单性上有积极的作用。整个数据库（定义、表、索引和数据本身）都在宿主主机上存储在一个单一的文件中。它的简单设计是通过在开始一个事务的时候锁定整个数据文件而完成的。

8.3.2 本地数据存储

离线应用抽象来说一般包括两种场景，即离线查询和离线更新。如果需要实现离线查询，那么需要将服务器端的数据下载到本地，对于 SaaS 应用而言则应该只下载当前租户的系统和业务数据。如果只实现离线更新，那么需要下载更新的业务对象涉及的关联数据，如客户、币别、部门等公共基础资料。

离线应用通过本地数据存储减少对服务器端的数据访问，从而提高应用性能，减少服务器压力，同样也能节省服务器资源。很多系统性能问题是由于大量查询请求造成的，并不是由数据添加造成的。数据同步模块在后台运行，默默地将本地数据库与服务器数据库进行数据同步。在大多数情况下，查询并不需要当前及时数据，数据有稍微的延迟并不会影响应用。对于本地添加修改的数据应该立即更新到本地数据库。

应用案例：一个销售管理系统，其业务会包括对订单数据的统计。订单主要包括以下这些数据项：

订单号	日期	商品	数量	价钱
1001	2008-10-15	笔记本	1	6000

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

如果对订单数据进行统计，一般都会有如下查询条件：查询某个时间段每天（或每月）销售的总额及柱状图。因此，我们需要以日期作为查询条件，并统计出每天（或每月）的总额。

例如：查询 2008-10-01 日到 2008-10-21 日每天销售额的流程图。

- 如果本地数据与服务器数据是一致的，则用户立即就获得 2008-10-01 日至 2008-10-21 日的数据。
- 如果当前是 2008-10-20 日，则无论本地和服务器都没有 2008-10-21 日的数据。
- 如果时间已经到了 2008-10-21 日，且有其他人在中心数据库输入了新数据。则 A 用户再次登录本地应用，继续查询 2008-10-01 日至 2008-10-21 日的数据，此时有可能本地没有 21 日的数据。当然同步模块本身就在后台进行自动同步数据，但未必会这么及时，用户稍等一会也应该是可以接收的。如果用户很紧急，可以提供同步进度界面提醒用户，以此提高用户体验。

8.4 数据同步

由于离线应用中数据在服务器中有一份，在本地数据库中也有临时更新的数据和查询的数据。同时离线使用会有时间间隔，多用户的数据提交很容易出现**冲突**问题。

数据同步将是整个系统关键的核心模块。该模块需要解决如下问题：

- 增量数据处理；
- 多用户冲突处理策略。

8.4.1 增量数据处理

我们可以采用 SVN 等版本管理软件以“数据版本”来识别增量的思路。我们这里将“数据版本”修改为“时间戳”更容易理解。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

服务器端的数据表：

订单号	日期	商品	数量	价钱	服务器端修改时间
1001	2008-10-15	笔记本	1	6000	2008-10-01 10:30:30

客户端的数据表：

订单号	日期	商品	数量	价钱	服务器修改时间	修改标记
1001	2008-10-15	笔记本	1	6000	2008-10-01 10:30:30	True/False

时间戳真实的含义：是一个数据的历史版本号。

服务器端在“时间戳”形式上的可选方案：

- **修改时间**：以具体时间 GMT 来表示数据的版本。
- **版本号**：即版本数字。

我们应该选择**修改时间**作为服务器端的**时间戳**。因为使用**修改时间**有个好处，即它不仅能代表数据的版本，同样也能代表数据具体修改时间点。而且，一般应用都会对数据的修改记录其**修改时间**，为了适应离线使用就不需要再添加字段了，直接使用**修改时间**作为**时间戳**。

客户端也需要一个时间戳：

- **修改时间**：以具体时间 GMT 来表示数据的版本。
- **版本号**：即版本数字。

无论是修改时间还是版本号都不能使用本机时间。因为在离线状态下，本机时间是会变化的，并不能与服务器时间一致；如果服务器与客户端都使用**版本号**作为时间戳就会有冲突。因为版本号基本采用递增方式，服务器从 1 递增到了 2，而离线用户在本地也是从 1 递增到 2。那同步时采用哪边呢？

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

因此，我们不能简单地在本地使用一个时间戳就可以了。我们需要将记录服务器端的时间戳记录到本地，再配合一个更改标记（bool 型）即可。而这个更改标记就是“本地时间戳”。

图 8-3 描述了增量数据的同步过程。如果本地记录的“服务器时间戳”与服务器端的“时间戳”一致，则说明从上次同步到现在服务器端数据未变化过；并且本地修改标记为 True。这两个条件就能说明本地数据要比服务器端的数据新，需要将本地数据更新到服务器。

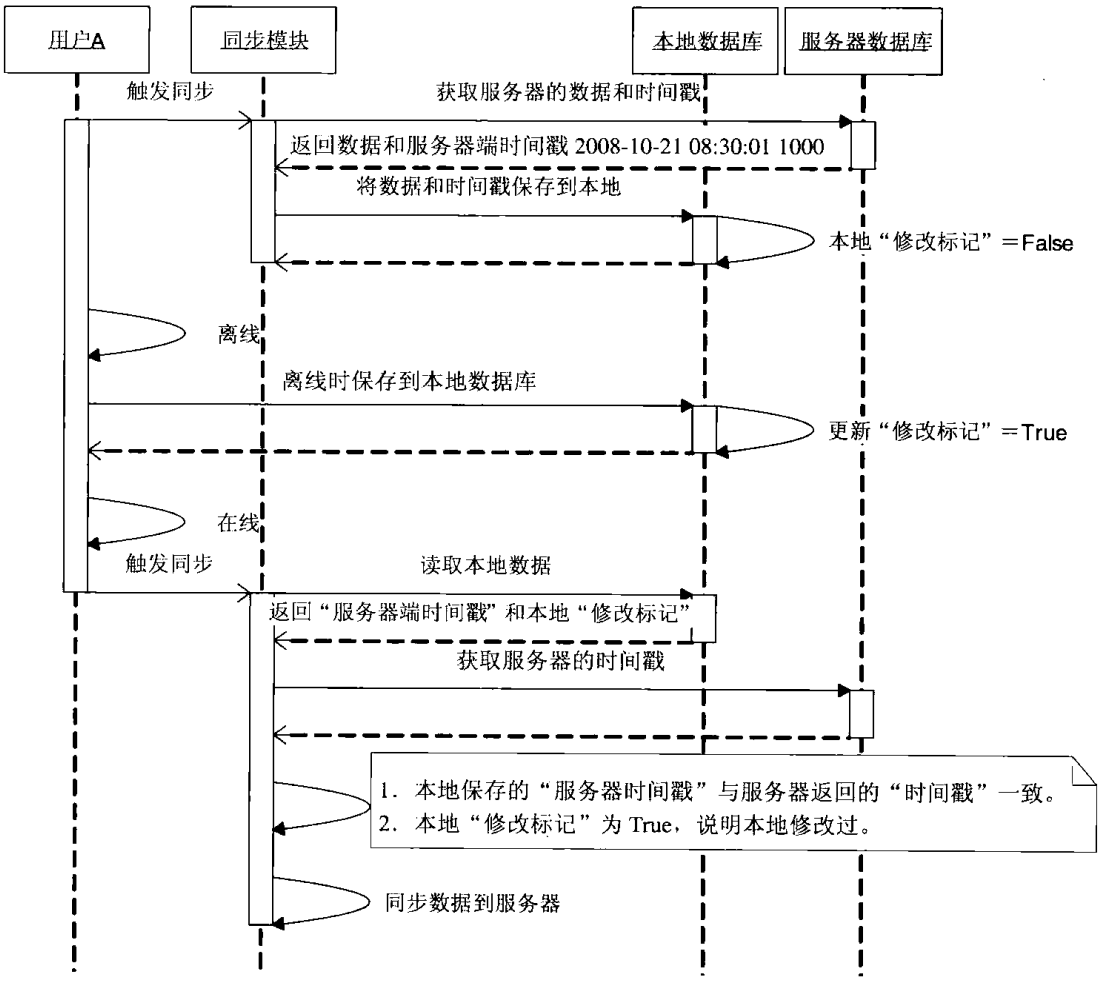


图 8-3　增量数据同步图

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

如果本地的“修改标记”为 False；而服务器端的时间戳比客户端保存的“服务器端时间戳”大。这两个条件就能说明服务器端数据要比客户端的数据新，需要将服务器端数据同步到本地。

8.4.2 多用户冲突

如果本地的“服务器时间戳”与服务器端的“时间戳”不一致，则说明有其他人修改了服务器的数据，需要更新到本地。如果本地**修改标记**也是 True，则会出现多用户冲突问题（如图 8-4、图 8-5 所示），需要对多用户冲突进行处理。

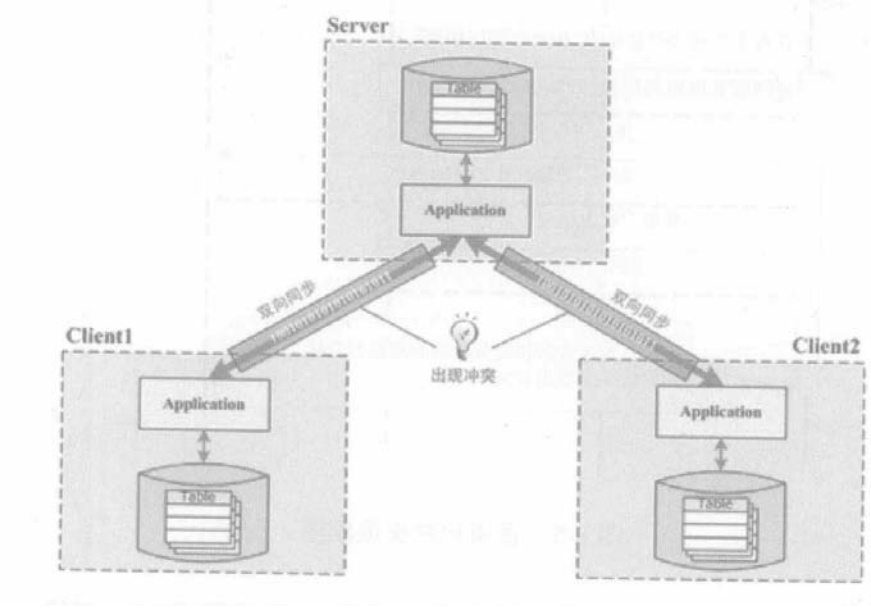


图 8-4 多用户冲突图

（1）冲突解决策略

服务器无法判断到底是用户 A 还是用户 B 的真实修改时间，因为真实修改时间并不是提交时间，真实修改顺序也不是提交顺序。对于服务器来说后提交者就是后修改者。因此离线应用在数据冲突方面可以考虑如下几种策略：

- **覆盖策略**：就是“后提交者”覆盖“前提交者”的内容，以“后提交者”内容为最终结果。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

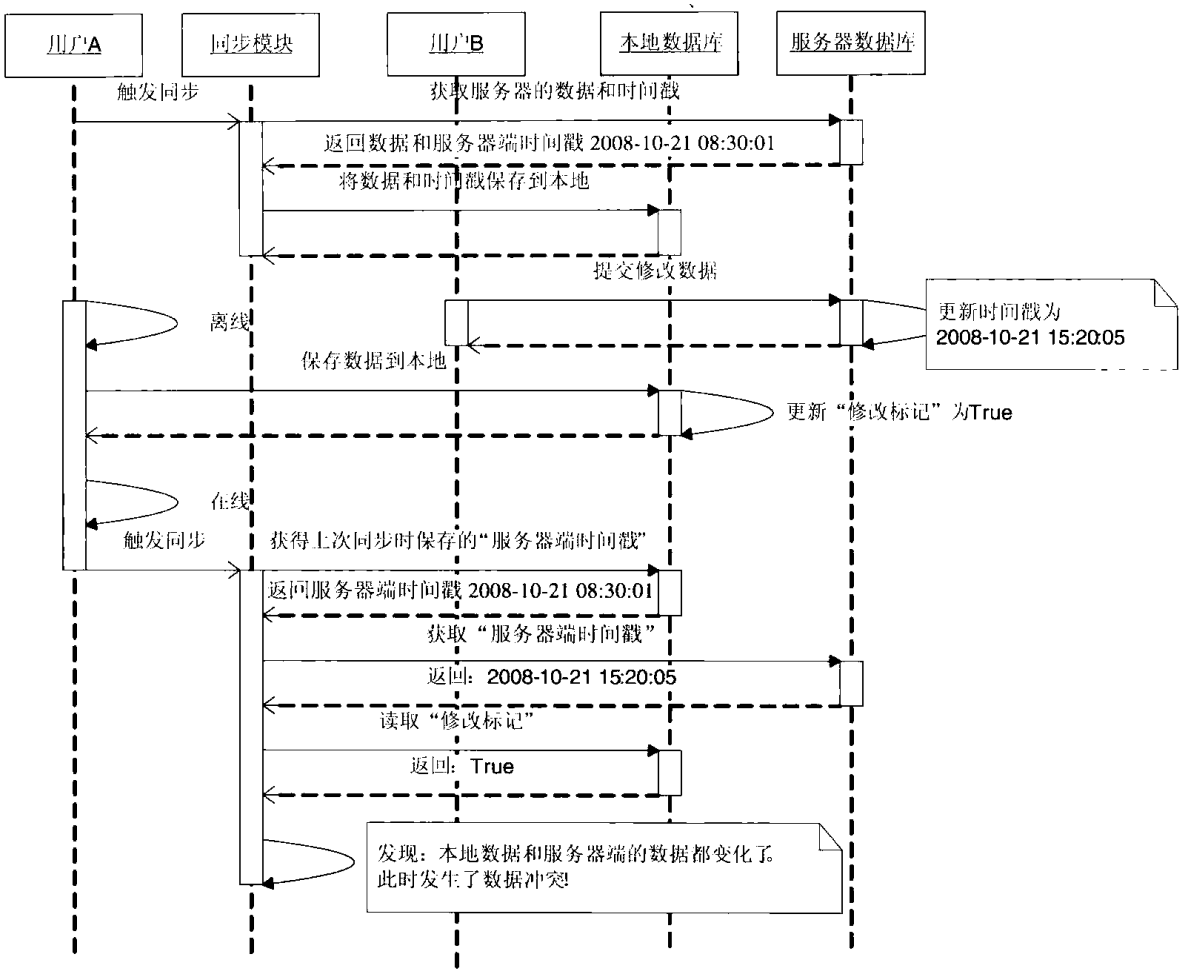


图 8-5　多用户冲突流程图

- **丢弃策略**：“后提交者”在提交前发现之前有人已经提交过，则将自己的数据丢弃，在系统里仍然保留“前提交者”的内容。
- **提醒方案**：当发生冲突时，则将冲突信息提醒用户，由用户分析后确定采用“覆盖”或“丢弃”。

在整个数据同步过程中冲突解决是个核心问题，而且也很复杂，每个应用采用的具体策略也不尽相同。本章主要给读者提一下相关的冲突场景和相关策略。

（2）数据同步次序

上面只给大家介绍了单条数据的同步问题。但很多应用是有很多表的，并且这些表

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

都互相关联（即存在外键）。因此同步次序需要按业务对象的依赖次序来上传和更新数据。对于数据可以采用如下 3 种方式打包同步：

- 所有变更数据打包；
- 按业务对象打包；
- 每条记录打包。

8.5 数据传输

在数据传输过程中，我们需要充分考虑网络带宽、数据安全、网络的不稳定性和后台处理等因素。对于这些因素，在技术上可以采用如下几个关键策略：

- **传输协议**：可以采用 HTTP、FTP 协议，也可以自己通过 Socket 实现自定义协议。
- **压缩**：考虑到网络带宽限制，我们需要将数据进行压缩处理，这能极大降低网络开销，一般多使用 gzip 解压缩。当然前面 8.4.1 节的增量数据处理也对网络的开销起到很关键的作用。
- **加密**：为了保障应用数据的安全性，必须要考虑对数据进行加密。比如采用 SSL 连接、数字签名等。
- **分包处理**：前面我们提到有 3 种打包形式，即按所有变更数据打包；按业务对象打包；按每条记录打包。
 - ✧ 按所有变更数据打包：可能数据包会很大，我们应该将其分成小包或文件进行传输。这时，就需要对这些小包或文件进行排序处理。这种打包形式在处理性能上并不是最优的，一次性提交的数据量大而且传输的次序控制也比较麻烦。
 - ✧ 按业务对象打包：将本地的修改数据按业务对象区分后单独打包。这样对于数据完整性是有所保障的，同时数据量也不会很大。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- ✧ 按每条记录打包：这种方式需要在本地对记录进行排序处理，因为表数据是有关联的。如果本地不处理而放到服务器端处理，则服务器端需要作对应的业务处理，客户端在提交数据时先保存数据，然后按表的关联顺序插入数据库。

- **断点续传**：由于网络的不稳定，对大数据包的上传会造成失败，从而无法完成数据的传输，因此需要实现断点续传功能。

在 3 种打包形式中，对于按所有变更数据打包的形式，使用断点续传能起到一定作用；对于按记录打包，则无须断点续传；如果采用分包处理也可以不需要断点续传。断点续传主要是针对单个大文件的处理才能起到优化的效果。

8.6 离线应用开发工具介绍

在网络应用的开发者群体中，目前正火热地议论着“离线应用”这个概念，以及如何使用离线应用技术。“离线应用”是 Google（谷歌）推出的一种使用技术，Google 将向开发者提供谷歌离线应用开发工具（Google Gears），这是一个为生成离线网络应用软件而开发的开源技术。

下面详细介绍 Google Gears 的特性，以帮助大家实现离线应用。

Gears 的三大核心模块：

- 本地数据库：此模块用于存储和访问用户在离线情况下提交的数据。
- 本地服务器：此模块用于存储网络资源文件，以便在离线情况下访问网页。
- 工作者池：这是一个提高 JavaScript 工作效率和页面性能的模块，使 JavaScript 代码像后台进程一样运行而不影响用户访问网页。

创建 Gears 对象：

```
(function()
{ …
  var factory = null;
  try {
```


互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        factory = new ActiveXObject('Gears.Factory');
    }catch (e) {
        ...
    }

    if (!factory) {
      return;
    }

    if (!window.google) {
      google = {};
    }

    if (!google.gears) {
      google.gears = {factory: factory};
    }
  })();
```

8.6.1 本地数据库

dbquery.html 工具实现了对本地 SQLite 数据库使用的简单例子。

```
//创建数据库操作对象
var db;

try {
    db = google.gears.factory.create('beta.database');
} catch (ex) {
    // Gears 没有安装
    alert('不能创建数据库:\n\n' + ex.message);
    showLoginForm();
    return;
}
//打开本地数据库
db.open(databaseName);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

8.6.2 本地服务

这一节涉及 Gears 技术的核心模块之一“本地服务器”，讲述如何在线捕获网页资源，在离线状态下可以随意访问已捕获的资源。

如下方法用于在内部创建 local server，并打开 local store。

```
var STORE_NAME = 'helloworld-store'

function openLocalStore() {
  if (window.google && google.gears) {
    try {
      //创建 local server
      localServer = google.gears.factory.create('beta.localserver');

      //创建 local store
      store = localServer.createManagedStore(STORE_NAME);
    } catch (ex) {
      //显示错误信息
      return;
    }
  }
}
```

capture ()方法用于在内部捕获 JSON 文件里指定的网络资源。

```
var MANIFEST_FILENAME = 'my_manifest.json';

function capture() {
  //确保 local store 已经创建
  if (store) {
    // JSON 文件里指定了我们需要的资源文件，读取它并捕获这些文件
    store.manifestUrl = MANIFEST_FILENAME;
    store.checkForUpdate();
  }
}
```


互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

删除 local store，用户只能在线使用（Gears 会防止离线情况下本地文件丢失）。

```
function removeStore() {
  if(localServer){
    if (localServer.openManagedStore(STORE_NAME)) {
      //删除 local store
      localServer.removeManagedStore(STORE_NAME);
      store = null;
    }
  }
}
```

当你完成自己的应用后，测试步骤如下：

- 启动 Apache，访问你的应用主页。
- 使用 capture 功能，使应用系统能够离线工作。
- 让网络连接断开（如果是单机版操作，即服务器和客户是同一台机器，可以将 Apache 服务器关闭来模拟网络连接断开的场景）。
- 为了确保不是由于浏览器缓存而依然能够访问的页面，我们先把页面缓存数据全部清除，如“删除 Cookies”、“删除文件”、“清除历史记录”。
- 此时，你处于 offline 的状态（与服务器通信失败），并且 IE 缓存也已经清除；接下来我们再试着与服务器通信访问这个页面。
- 重新打开一个新的浏览器，输入该网页的 URL，访问它。访问是成功的！因为 Gears 已经把 my_menifest.json 里指定要捕获的网络资源保存到本地的存储源里了，离线时即可从本地存储源中加载。浏览器的“工具”选项里有一个“Gears 设置”选项（见图 8-6），你可以将 Gears 存储的那些访问过的网址资源清除。清除后，你将无法离线访问该网址。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

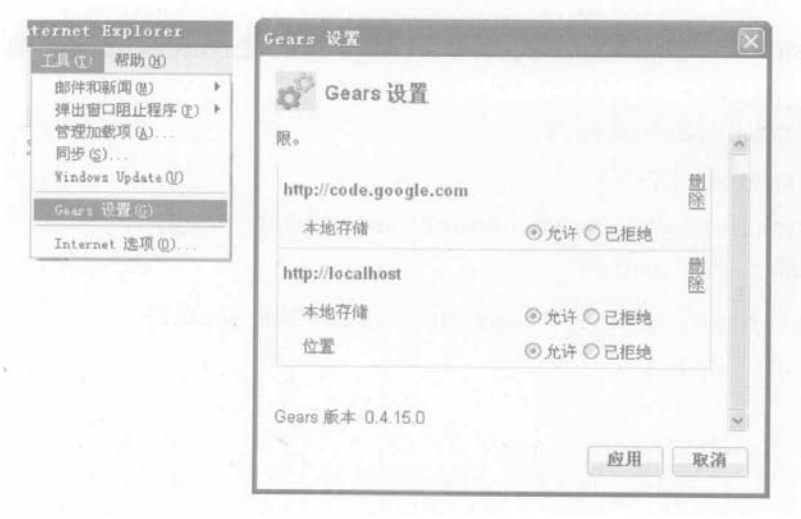


图 8-6　Gears 设置

8.6.3　工作者线程池

如果是离线状态，则使用 Gears 的本地数据库暂存数据；当网络连通时，需要将数据传输到服务器上，这就需要 Gears 的工作者线程池模块。

```
var workerPool;
workerPool = google.gears.factory.create('beta.workerpool', '1.0');
// 创建子 worker
childId = workerPool.createWorkerFromUrl('js/worker.js');

//用户单击 sync 按钮将触发子 worker 的工作
function asyncExc() {
    if (workerPool) {
        var message = 'sync';
        workerPool.sendMessage(message, childId);
    }
}
```

在 worker.js 文件中添加如下代码：

```
var db;
var request;
```


互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    function childInit() {
    request = google.gears.factory.create('beta.httprequest','1.0');
    db = google.gears.factory.create('beta.database', '1.0');
    google.gears.workerPool.onmessage = childHandler;
    }

    function childHandler(msg, sender) {
      if(msg=='sync') {//if the message recieved from parent is 'sync'
        rs = db.execute('select date,money,content from bills order by date
desc');
        var date;
        var money;
        var content;

        while (rs.isValidRow()) {
            date = rs.field(0);
            money = rs.field(1);
            content = rs.field(2);

            //同步操作
            var url = '../action.jsp?date='+date+'&money='+money+'&content=
'+content;
            request.onreadystatechange = uploadCallback;
            request.open('GET',url, true);
            request.send(null);

            //删除
            db.execute('delete from report where name=? and title=?
    and content=? and Timestamp=?', [name,title,content]);
              rs.next();
            }
            rs.close();
      }
    }
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
//回调函数，什么也不做
function uploadCallback() {
  if(request.readyState == 4) {
    if(request.status == 200) {
      //你可以在这添加需要的代码
    }
  }
}

childInit();
```

8.6.4 Gears 待发展的内容

通过对 Gears 的分析，我们发现它提供了本地数据库、本地服务、工作者线程池等为离线应用开发所需的工具。但对于一个离线应用系统不仅仅需要解决这些问题，它还需要解决增量数据同步、数据冲突处理、数据传输问题，如多用户冲突问题的策略选择模块，服务器端配合的数据同步模块等。对于这些也是大多数离线应用需要共同面对的问题，因此 Gears 也可以针对这些通用问题继续研究和开发，让广大应用开发者共同收益。

8.7 小结

在 Saas 应用中采用离线应用需要解决如下 4 个系统性问题：

- 本地服务；
- 本地存储；
- 数据同步；
- 数据传输。

对于这样的应用能解决用户在离线状态下的使用问题，同时在网络不稳定的情况下也能避免正在编辑的数据丢失。离线应用难度最大的是如何处理同步，这部分也是本章的核心内容。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第3篇

SaaS 基础架构——云计算

今天的互联网业界对云计算的关注程度，已经不亚于对一次革命。微软、谷歌、Salesforce、Amazon 等国外软件巨头纷纷推出了自己的云计算系统，意图领导这一轮的软件新浪潮。国内的阿里巴巴也已经投入大量资源进入云计算这片新的海洋。

基于互联网的软件系统需要承担海量的数据、文件、数据库里的数据，等等。这就必然需要可扩充的、廉价的、可靠的和高性能的网络硬盘来提供图片、视频、博客等文件的存储，还需要可扩充的、廉价的、可靠的和高性能的分布式数据库系统来提供业务数据的存放和获取，以及高速的海量内存 Cache 系统来对部分数据进行缓存，以减少业务系统的响应时间。

云计算系统可以解决以上需求，但不只是这些技术。那么云计算的这些技术如何实现？为什么会在互联网环境中提出云计算？云计算系统能否通过开源的软件组建呢？

下面的章节，将分别就分布式文件存储、分布式数据库、分布式 Cache 机制和分布式计算这几个方面进行阐述。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第9章 分布式文件存储

9.1 大文件的分布式存储 ……195
9.1.1 GFS ……195
9.1.2 HDFS ……197
9.2 小文件的分布式存储 ……202
9.2.1 ADFS（Alibaba Distributed File System） ……203
9.2.2 小文件存储的另外方案介绍 ……208
9.3 小结 ……209

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在前面的 CRM 例子中，根据客户和市场需要增加了邮件服务，邮件服务需要存储很多邮件文件在服务器里。同时，该 CRM 系统需要对用户的操作日志进行记录，并需要保存半年甚至一年以上，因此也对日志文件的存储空间要求提出了挑战。

目前企业在存储超过 1TB（1024GB）以上数据文件时，一般是采用 DAS/NAS/SAN 架构。此类架构存储几十 TB 以下数据还可以承受，但成本很高，比如：

- 采用 NAS，一般搭一套 2TB 左右的系统，就需要花费 100 万左右；
- 若使用 SAN 架构，成本还会更高点。若要搭 10TB 左右，成本则需要几百万。

而若要以 PB（1024TB）为单位，目前是无法做到的，或者成本异常高。由于互联网的数据是海量的，一个大的网站的数据就可以达到并超过一个 PB（1024TB），因此传统的 NAS/SAN 方式是不能满足需要的。

9.1 大文件的分布式存储

为了存储更大量的数据，需要改变原有思想，采用新的架构。目前有两个产品：GFS 与 Hadoop 的 HDFS。

9.1.1 GFS

GFS 是 Google 的分布式文件系统解决方案。GFS 的设计目标是解决以 PB 为单位的海量存储问题。它采用分布式存储，多份拷贝的技术架构，只使用普通服务器加普通硬盘的方式。

GFS 的设计约定如下：

（1）系统是建立在大量廉价的普通计算机上的，而且这些计算机可能容易出现故障，因此系统本身必须进行自我检测和恢复。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（2）系统存储许多大文件。系统预计存储几百万文件，每个文件一般是 100MB 或更大，几个 GB 的文件经常出现，而且能高效地管理。小文件也必须支持，但不做特别优化。

（3）系统工作情况一般分为两类读：一类是大部分的流式读，另一类是少量的随机读。流式读的数据一般超过 1MB 甚至更多，少量的随机读一般只读几 KB 的数据。

（4）系统工作中写操作经常是许多大的串行写数据到文件。一旦写后，很少有更改操作。支持文件中任意位置的写操作，但不必高效。

GFS 由单个 Master 服务器和多个块服务器组成。GFS 中的文件由块组成，块一般以 64MB 为单位。块由 Master 分配管理，每个块有几个拷贝（默认是 3 个），分别在不同的块服务器上。文件到块的映射关系全部放在 Master 的内存中。Master 和块服务器之间定期进行通信，块服务器定期报告信息给 Master。当 Master 重启时，从所有的块服务器取回块的信息。文件是按照普通的目录分层方式组织的，系统实现与 Posix 类似的 open、write、read、close 等函数。

GFS 的设计框架如图 9-1 所示。

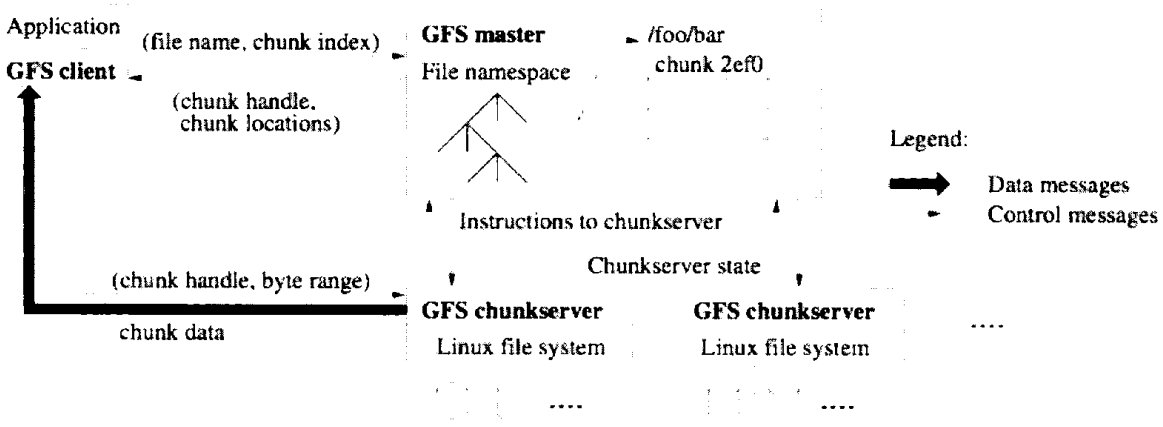


图 9-1　GFS 系统架构

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

关于 GFS 的更详细介绍，请参考 Google 的论文：http://labs.google.com/papers/gfs.html

9.1.2 HDFS

HDFS 是基于 Apache 基金的 Hadoop 的核心分布式文件系统，也是在 Google 的 GFS 思想基础上开发的开源系统，它实现了 GFS 的绝大部分功能。

Hadoop 包含两类技术：

- 一是分布式文件存储系统（HDFS）；
- 二是 MapReduce 分布式计算（参见第 12 章）。

Hadoop 的技术框图如图 9-2 所示。

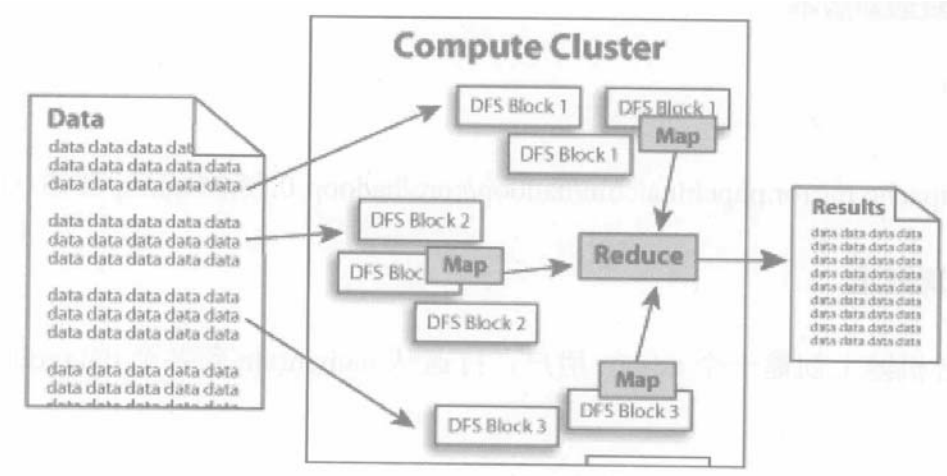


图 9-2 Hadoop 系统架构

HDFS 的文件由块组成，而且每个块都有多个副本，每个副本存放在集群中的不同计算机上。因此，MapReduce 可以把应用分割成许多小块的工作单元，每个工作单元（就是图 9-2 中的 Map）处理它可以访问的副本，并把数据处理结果汇集（Reduce）起来。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

由于 Hadoop 是按照 GFS 的设计思想开发的开源系统，因此我们可以利用 Hadoop 来部署类似 GFS 的分布式文件系统。Hadoop 可以从 Apache 的网站上下载，下载地址是 http://hadoop.apache.org/，目前最新版本是 0.18。

下面重点介绍如何基于 Hadoop 进行部署和开发。

1. 准备硬件

使用 4 台服务器，其中 1 台作为 namenode，3 台作为 datanode，机器名分配如下：

```
10.2.224.24 namenode
10.2.224.25 datanode1
10.2.224.26 datanode2
10.2.224.27 datanode3
```

2. 下载最新版本

地址是：

http://apache.mirror.phpchina.com/hadoop/core/hadoop-0.18.0/hadoop-0.18.0.tar.gz

3. 环境搭建

在每台机器上创建一个 admin 用户，打通从 namenode 到各个 datanode 的 ssh 通道；

```
mkdir /home/admin/hadoop-installed
```

在/home/admin/hadoop-installed 目录下解压 hadoop 压缩包，解压为 hadoop。

（1）配置集群模式

```
mkdir /home/admin/hadoop-installed/cluster-conf
```

在 cluster-conf 目录下创建如下 4 个文件：

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- masters

```
namenode #master 机器的机器名
```

- slaves

```
datanode1
datanode2
datanode3
```

- hadoop-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/admin/hadoop-installed/filesystem</value>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://namenode:54310</value>
</property>
<property>
<name>mapred.job.tracker</name>
<value>hdsf://namenode:54311</value>
</property>
<property>
<name>dfs.replication</name>
<value>3</value>
</property>
<property>
<name>mapred.child.java.opts</name>
<value>-Xmx512m</value>
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    </property>
    </configuration>
```

- hadoop-env.sh

```
    export JAVA_HOME=/usr/lib/j2sdk1.5-sun
    export  HADOOP_NAMENODE_OPTS="-Dcom.sun.management.jmxremote  $HADOOP_
NAMENODE_OPTS"
    export   HADOOP_SECONDARYNAMENODE_OPTS="-Dcom.sun.management.jmxremote
$HADOOP_SECONDARYNAMENODE_OPTS"
    export  HADOOP_DATANODE_OPTS="-Dcom.sun.management.jmxremote  $HADOOP_
DATANODE_OPTS"
    export  HADOOP_BALANCER_OPTS="-Dcom.sun.management.jmxremote  $HADOOP_
BALANCER_OPTS"
    export HADOOP_JOBTRACKER_OPTS="-Dcom.sun.management.jmxremote $HADOOP_
JOBTRACKER_OPTS"
    export HADOOP_LOG_DIR=/home/admin/hadoop-installed/logs
```

（2）增加环境变量 HADOOP_CONF_DIR

```
    export HADOOP_CONF_DIR=/home/alisoft/hadoop-installed/conf
```

（3）配置单机调试模式

增加 single-conf 目录，在该目录下也增加如下 4 个文件：

- masters

```
    localhost #namenode 机器的机器名
```

- slaves

```
    localhost
```

- hadoop-site.xml

```
    <?xml version="1.0"?>
    <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
```


互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/admin/hadoop-installed/filesystem</value>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:54310</value>
</property>
<property>
<name>mapred.job.tracker</name>
<value>hdsf://localhost:54311</value>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>mapred.child.java.opts</name>
<value>-Xmx512m</value>
</property>
</configuration>
```

- hadoop-env.sh

```
export JAVA_HOME=/usr/ali/java
export HADOOP_NAMENODE_OPTS="-Dcom.sun.management.jmxremote $HADOOP_
NAMENODE_OPTS"
export HADOOP_SECONDARYNAMENODE_OPTS="-Dcom.sun.management.jmxremote
$HADOOP_SECONDARYNAMENODE_OPTS"
export HADOOP_DATANODE_OPTS="-Dcom.sun.management.jmxremote $HADOOP_
DATANODE_OPTS"
export HADOOP_BALANCER_OPTS="-Dcom.sun.management.jmxremote $HADOOP_
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
BALANCER_OPTS"
    export HADOOP_JOBTRACKER_OPTS="-Dcom.sun.management.jmxremote $HADOOP_
JOBTRACKER_OPTS"
    export HADOOP_LOG_DIR=/home/alisoft/hadoop-installed/logs
```

现在你就可以简单地切换集群模式和单机调试模式了。

```
    ln -s cluster-conf/ conf 切换到集群模式
```

或者

```
    ln -s single-conf/ conf 切换到单机调试模式
```

Hadoop 提供了一系列命令操作接口，用户可以使用命令行来操作 Hadoop 文件系统。例如：

```
    ./bin/hadoop fs -ls /home/alisoft/hadoop-installed/filesystem/mapred/
```

查看文件目录，详细的命令行请参考 Hadoop：

http://hadoop.apache.org/core/docs/current/commands_manual.html

http://hadoop.apache.org/core/docs/current/hdfs_shell.html

9.2 小文件的分布式存储

因为 GFS/Hadoop 是使用单 Master 模式，把所有的文件与块之间的关系放在内存中，而且设计的目的是用于存放大文件（大于 100MB），这样对于有海量（几十亿）的小文件（如几十 KB 至几百 KB 的图片）则是无法支持的。因此，单 Master 模式不可能把所有的文件与块之间的关系都存放到内存中。

CRM 系统中的邮件功能，邮件文件需要长期保存在服务器上，这样当用户数达到一定程度后，存储在服务器上的文件数量将是海量的，如以亿为单位。另外，邮件文件的平均大小都不大，一般一个邮件文件都小于 100KB。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

邮件系统的特性决定了我们不能使用类似 GFS 的文件系统来存储邮件文件。因此，需要设计一套新的文件系统来解决海量小文件的存储。

9.2.1 ADFS（Alibaba Distributed File System）

针对海量文件，需要修改 GFS 的设计模式来加以支持。假设：

（1）系统中存在海量的文件，文件的数量是以亿为单位的。

（2）大多数文件是小文件，一般小于 1MB（最大的为 2GB）。

（3）文件通常是一次写、多次读（写了之后很少有修改，当然系统支持对文件进行修改操作）。

为了解决上述问题，我们设计了一个 ADFS（Alibaba Distributed File System，阿里巴巴分布式文件系统）。

如何解决上面的问题，我们需要对 GFS 设计进行修改，如下：

（1）修改文件和块的关系，允许一个块内存放多个文件。

（2）修改单 Master 模式，使用多 Master 模式来存放文件和块的对应关系。

（3）使用分布式内存 Cache 系统来缓存文件和块的对应关系。

我们首先设计新的文件块格式。我们对文件与块的对应关系进行修改，增加块内偏移（offset），即一个文件和块的对应关系为：

文件名⇔（块 ID+块内偏移）+{中间块 ID}+（块 ID+块内偏移+块内长度）

{中间块 ID}代表 0 或 N 个块 ID；块内长度是以页为单位的（默认 1 页为 16KB），每个页有一定长度的页头用来描述该页信息，假设为 20 字节。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

文件和块的对应关系图如图 9-3 所示。



图 9-3　文件和块的对应关系图

我们举两个例子来说明这个文件块格式。

例 1：假设一个文件的大小为 200MB，块的大小设定为 64MB，那么这个文件至少占用 4 个块的空间。假设有 4 个空的块，编号是 1001、1002、1003、1004、那么这个文件对应的块编码信息为：（1001,0）（1002）（1003）（1004,0,515）。这个文件和块的信息存储在 Master 的存储系统中。

例 2：假设一个文件的大小为 100KB，由于一个快的大小设定为 64MB，因此这个文件只要一个块就可以存储了。为此 Master 找到一个空闲空间超过 100KB 的块。假设块 ID 为 1102，该块已经存放了几个小文件了，还剩余 100 页的空间（$100 \times 16=1600$KB），也就是说，在第 3996 页开始是空闲的。那么，我们把这个文件头放在第 3996 页上，占用连续余下的 6 页空间（100/16+1=7，这里忽略了页头占用的 20 字节大小空间）。因此，该文件的块描述为（1102,3996,7）。

■ 修改 Master 设计模型

由于文件和块的关系数量巨大，达到亿的级别，已经不能全部放在内存中了。因此，简单的存储已经不能满足需要，需要使用多台数据存储服务器来存储文件和块的关系。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

把映射关系数据和逻辑分配管理系统分离。逻辑分配管理仍旧是原来的 Master 系统，但原有的 Master 系统内不再存储文件和块的映射数据，只负责分配新的文件分配块，以及文件副本增加、减小。也就是说，只负责文件和块映射关系的更改操作，不负责查询操作。查询从 Master 中分离出去，由数据管理系统提供查询功能，简单的系统架构图如图 9-4 所示。

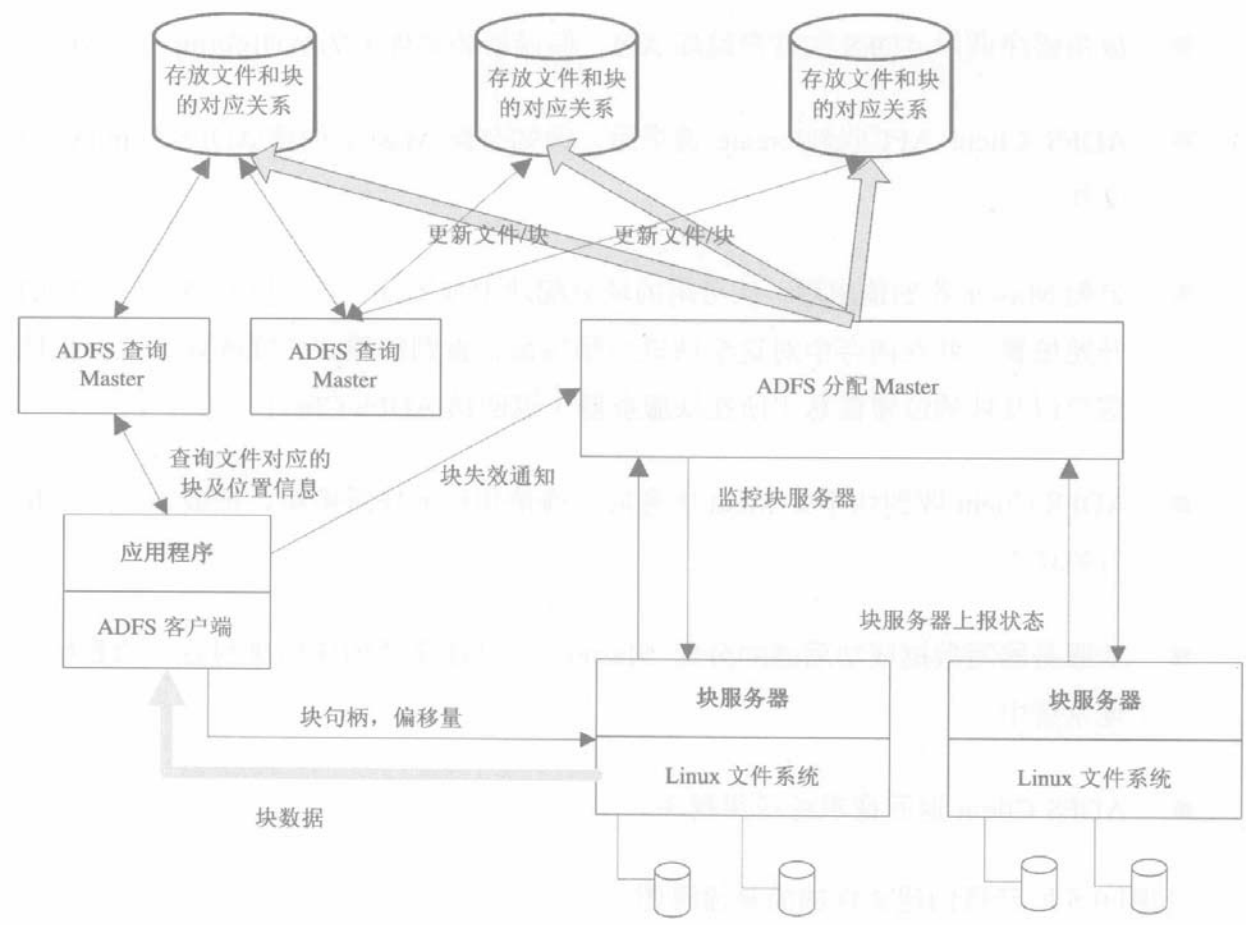


图 9-4　ADFS 文件系统架构

从图中 9-4 可以看出，当 ADFS 文件系统应用程序希望读取某个文件数据时，它不是直接访问 ADFS 主分配 Master，而是访问查询 Master。由查询 Master 返回该文件及 Meta 信息和块相关信息，降低分配 Master 的压力。由于分配 Master 只在文件更改（创

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

建、删除、更新、副本增加、减少）时才处理业务，而系统中的操作绝大多数是读操作而非写操作，因此通过分配和查询的分离，大大提高了分配 Master 的处理能力。

下面介绍两个主要的操作流程：创建文件和读文件。

创建文件流程如下：

- 应用程序调用 ADFS 的客户端库 API，假设新的文件名为/ADFS/tmp/test.txt；
- ADFS Client API 收到 create 请求后，通知分配 Master 创建/ADFS/tmp/test.txt 文件；
- 分配 Master 收到请求后，从可用的块分配池中取出几个块，以及每个块空闲的开始位置，并在内存中对这个块进行预保留，直到创建文件完成或失败。把块信息以及块的位置信息（所在块服务器）返回给 ADFS Client；
- ADFS Client 收到块相关 Meta 信息后，连接相应的块服务器，把数据写入到相应的块中；
- 块服务器写数据成功后通知分配 Master，记录该文件的块信息到后台的数据管理系统中；
- ADFS Client 返回请求给应用程序。

如图 9-5 所示是创建文件的简单流程图。

读文件流程如下（流程图见图 9-6）：

- 应用程序调用 ADFS 的 Client API 库，读取/ADFS/tmp/test.txt 文件的数据；
- Client API 收到请求后，发出获取该文件 Meta 和块信息请求给查询 Master；

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

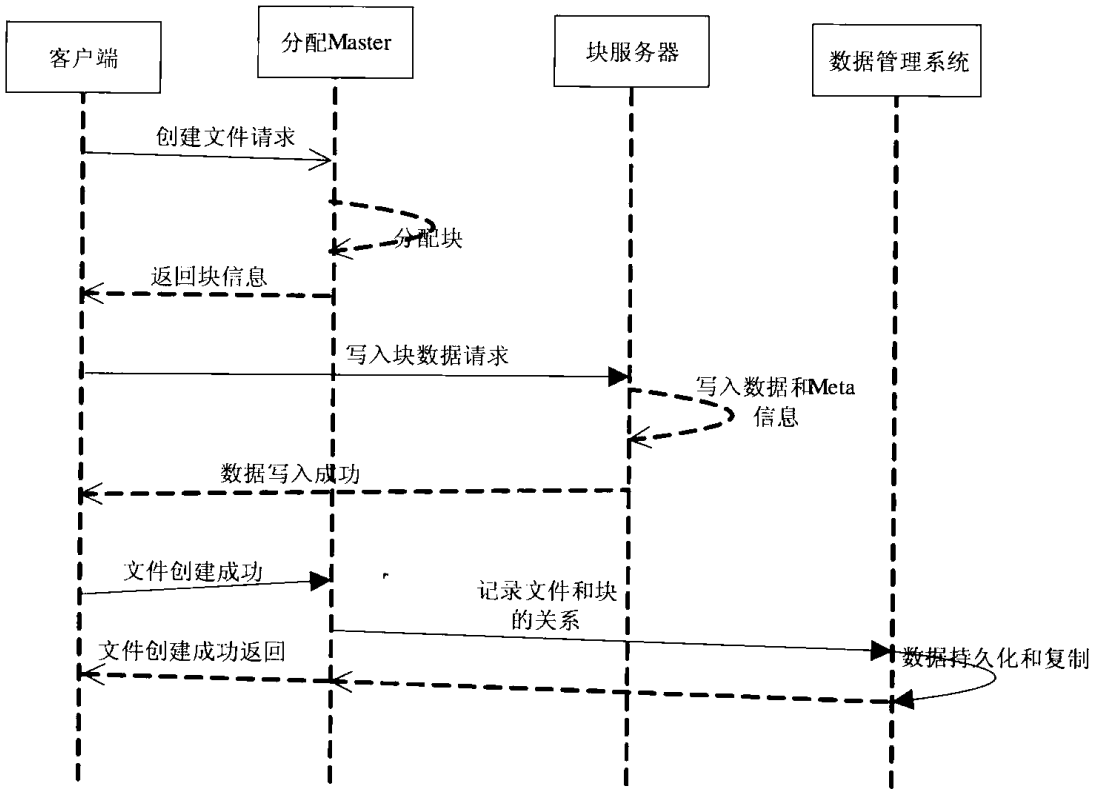


图 9-5　创建文件流程图

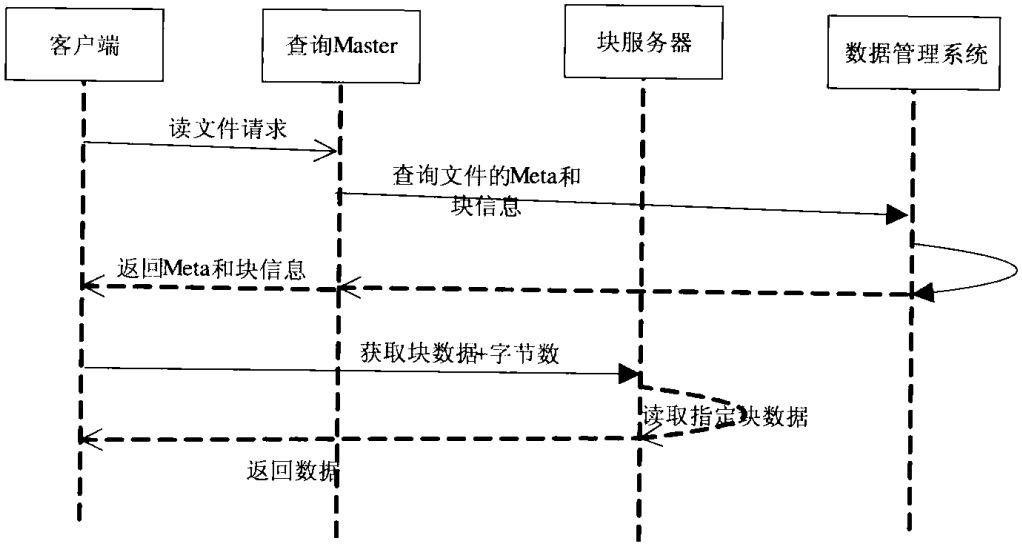


图 9-6　读文件流程图

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 查询Master向后台的数据管理系统查询该文件的相关信息，查到后返回给Client API；
- Client API 得到 Meta 信息和块的信息后，向对应的块服务器发出读相关的块数据命令；
- 块服务器根据指令读取相应的块数据并返回给 Client API；
- Client API 返回请求给应用程序。

9.2.2 小文件存储的另外方案介绍

ADFS 文件系统的设计目的是解决海量小文件问题。对于海量小文件，其实还有一个设计，就是不使用块模式，而是直接使用文件模式。

对每个文件设置多个副本，而不是对块设置多个副本。

使用文件模式的好处是整个设计简单化，在块服务器上存放的不是块，而是每个文件的副本。使用文件模式的缺点是每个块服务器上文件数过多。

我们假设平均每个文件为 100KB，每台块服务器的容量为 2TB。那么存放 2TB 的文件数目约为：$2 \times 1024 \times 1024 \times 10=20971520$ 个真实文件；而使用块模式（假设块的大小为 100MB），存放 2TB 的文件数目约为：$2 \times 1024 \times 10=20480$ 个块文件。这样，在使用块模式的情况下，块服务器可以把所有的块文件句柄存放在内存中，以减少打开文件所消耗的时间；而使用真实文件模式，那么这么多文件句柄是无法放在内存中的，只能通过 LRU 算法进行文件句柄的缓存。

目前，没有测试过这两种模式对于海量小文件的吞吐性能差别，有兴趣的读者可以自己测试一下。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

9.3 小结

本章简单地介绍了分布式文件存储系统的两个类型系统，分别适用于不同的应用场景。

- 对于视频文件、搜索引擎文件、大的日志文件等系统应用，使用类似 Google 的 GFS 的大文件系统比较合适；
- 对于存储海量小的图片文件之类应用，使用本章介绍的类似于 ADFS 之类的小文件系统比较合适。

SaaS 架构设计

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第 10 章 分布式数据库

10.1 基于列的结构化分布式数据库 ……211
10.1.1 HBase 数据库数据结构 ……211
10.1.2 HBase 系统架构 ……214
10.1.3 HBase 实践 ……215
10.2 基于代理的分布式数据库 ……217
10.3 基于列的分布式关系数据库 ……237
10.4 小结 ……241

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

随着互联网的发展，互联网上各种应用的数据越来越多，对数据库的处理能力要求越来越高。基于传统的以小型机/大型机作为载体的集中式数据库系统的成本越来越高，而且处理能力越来越无法满足应用需要。为此需要考虑使用分布式数据库系统来进行业务数据的存取。

在前面的 CRM 例子中，当租户数量达到一定程度，如 5000 时，使用单个 PC 服务器的数据库系统将无法承受业务数据的压力。对于 CRM 系统的服务提供商，从经济的角度考虑，需要采用分布式数据库系统来解决业务数据的增长带来的压力问题。

分布式数据库又可分为几类：基于列的结构化分布式数据库、基于代理的分布式数据库、分布式关系数据库。

10.1 基于列的结构化分布式数据库

前面第 5 章讲到的可配置性，其中对于定制列的数据存储，使用传统的关系型数据库来存储，性能不高，尤其当数据量非常大时，性能将急剧下降。本节介绍的内容可以解决该问题。

基于列的结构化分布式数据库有 Google 的 Bigtable、Aapche 的 HBase。HBase 与 Bigtable 类似，这里重点介绍开源的 HBase。

HBase 是一个结构化数据的分布式存储系统，是建立在 Hadoop 核心分布式文件系统 HDFS 基础上的可扩展的结构化数据库。

10.1.1 HBase 数据库数据结构

传统关系数据库的表结构是由行记录组成，而行记录又是由列字段组成，每行记录拥有的列字段数是相同的。假设有一张网页内容表（htmlcontent）：

传统数据库的表结构如表 10-1 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 10-1　htmlcontent 表结构

"URL" 段	"contents" 字段	"anchor" 字段	"mime" 字段
"com.cnn.www"		"anchor:cnnsi.com"	
"com.cnn.www"		"anchor:my.look.ca"	
"com.cnn.www"	<html>...		"text/html"
"com.cnn.www"	<html>...		
"com.cnn.www"	<html>...		

上面的表里有 5 行记录，对应的在物理存储上会存储 5 行记录。当应用程序查询

URL="com.cnn.www"的记录时，通过索引定位到 5 行记录，数据库系统将分别进行磁盘定位 5 次（离散方式），把数据从磁盘中读出。

另外，上面的 htmlcontent 表存储的每行数据都有固定的列，当一行数据增加一列时，将影响所有行的数据。

而 HBase 的设计思想与传统关系数据库不同，它以列的方式来存储数据，而不是行。

HBase 的表结构如表 10-2 所示。

表 10-2　表结构与行记录格式

行关键字	时间戳	列"contents:"	列"anchor:"	列"mime:"
"com.cnn.www"	t9		"anchor:cnnsi.com"	
	t8		"anchor:my.look.ca"	
	t6	"<html>..."		"text/html"
	t5	"<html>..."		
	t3	"<html>..."		

HBase 也是以表的方式存储行数据。但这里的“行”与传统数据库的行不一样，一行数据有一个排序的关键字以及任意多的列。表是松散存储的，只要你喜欢，同一张表的行数据可以有可变的列数；而传统数据库的每行数据拥有相同的列字段信息。一个列

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

是以“<列族>:<标签>”这样的格式表示的，<列族>和<标签>可以是任何字符串。表通常由一系列的行记录组成，每行记录由一个行关键字定位，拥有可变的列字段数。上面的表 10-2 中，在 HBase 里，从逻辑上讲，其实是 1 行数据，而不是 5 行。

HBase 把列族数据在物理上紧密地存储在一起，因此对一个列族中的数据可以用相同的读/写操作。写操作是行锁，不能同时锁定多行记录，所有行写操作默认是原子操作。因为行数据必须有一个关键字来定位一行记录，因此创建行数据、操作行数据时，必须指定一个行关键字，行关键字可以是任何字符串。

HBase 表数据是一个松散的行数据集。实际上，在物理磁盘上是按照每列数据的方式存储的，而不是按照传统数据库那样按照每行数据的方式存储。这一点是区别传统数据库的最大不同，也就是说，上面的记录在物理上不是 5 行记录，而是 6 行记录，见表 10-3、表 10-4、表 10-5。

表 10-3　列记录格式 1

行关键字	时间戳	列 "contents: "
"com.cnn.www"	t6	"<html>..."
	t5	"<html>..."
	t3	"<html>..."

表 10-4　列记录格式 2

行关键字	时间戳	列"anchor: "	
"com.cnn.www"	t9	"anchor:cnnsi.com"	"CNN"
	t8	"anchor:my.look.ca"	"CNN.com"

表 10-5　列记录格式 3

行关键字	时间戳	列"mime: "
"com.cnn.www"	t6	"text/html"

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

10.1.2 HBase 系统架构

HBase 的整体架构图如图 10-1 所示。

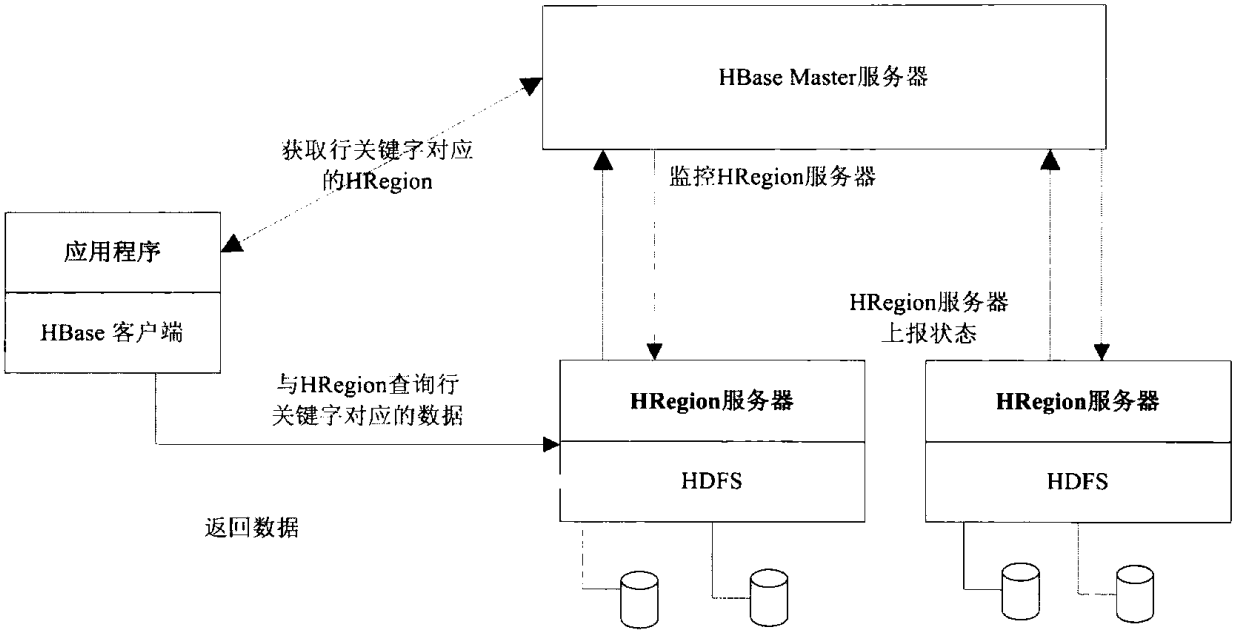


图 10-1 HBase 数据库系统架构图

即由 Master 和 Hregion（相当于 **Bigtable 中的 Tablet**）服务器组成。从用户角度看，一张表好像有一系列的排序行数据组成，物理上，表被分割成一系列的子表（HRegion）。一个 HRegion 由表名+一对“开始/结束”的行关键字唯一标识。一个 HRegion 存储所有行关键字在[开始]-[结束]范围内的行数据，一系列 HRegion 适当排序后组成了一张真正的表。所有数据物理上是存储在 HDFS 上的，提供数据服务给客户端的服务器叫做 HRegion 服务器，同一时刻，一个 HRegion 只能由一个 HRegion 服务器提供服务。当一个应用程序需要更新数据时，它连接到相应的 HRegion 服务器上，并提交数据更新给 HRegion，HRegion 服务器把提交的数据添加到 HRegion 的 HMemcache 和 HRegion 服务器的 HLog 中。

每个 HRegion 服务器由单个 HMaster 服务器管理，并定期与 Master 通信。HMaster

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

负责通知每个 HRegion 服务器它应该加载哪个 HRegion 并提供服务。HMaster 定期与 HRegion 服务器保持通信，若超时，那么 HMaster 和 HRegion 服务器将采取以下措施：

- HRegion 服务器将退出服务，并重新启动进入一个空闲状态；
- HMaster 会假设 HRegion 服务器已经死掉，将把原先的 HRegion 重新分配给另外的 HRegion 服务器。

10.1.3 HBase 实践

介绍了 HBase 的基本架构设计和概念后，我们举例说明如何开发基于 HBase 的应用。

首先是下载并安装 HBase，如何安装请参考 HBase 的官方文档，如 http://wiki.apache.org/hadoop/Hbase/10Minutes。

安装 HBase 需要准备一些环境，否则可能会失败，安装要求环境如下：

- Java 1.5.x 以上，最好是 Sun 提供的虚拟机；
- Hadoop 0.17.x；
- ssh/sshd 必须安装，而且 ssh 必须打通 Master 和子服务器之间的通道；
- HBase 是基于文件的，因此需要增大每个进程打开的文件句柄数；
- 集群里服务器的时间应该同步，最好使用 NTP 进行时间同步。

我们假设你已经部署并启动了 HBase 系统。假设你的程序是用 Java 开发的，而且已经创建了一张叫做“myTable”的表，此表有一个叫做“myColumnFamily”列族的列。

```
import java.io.IOException;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Scanner;
import org.apache.hadoop.hbase.io.BatchUpdate;
```


SaaS 架构设计

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
import org.apache.hadoop.hbase.io.Cell;
import org.apache.hadoop.hbase.io.RowResult;

public class MyClient {

  public static void main(String args[]) throws IOException {
    //初始化配置文件对象，告诉客户端如何访问 HBase
    HBaseConfiguration config = new HBaseConfiguration();

    // 初始化一个连接“myTable”表的 HTable 对象
    HTable table = new HTable(config, "myTable");

    // 初始化行关键字为“myRow”的批处理更新对象
    BatchUpdate batchUpdate = new BatchUpdate("myRow");

    // 设置列的值
    batchUpdate.put("myColumnFamily:columnQualifier1",
      "columnQualifier1 value! ".getBytes());

    // 删除这个列的数据
    batchUpdate.delete("myColumnFamily:cellIWantDeleted");

    // 提交 HBase 处理上面两个请求
    table.commit(batchUpdate);

    // 现在测试刚才更新的数据是否正常
    Cell cell = table.get("myRow", "myColumnFamily:columnQualifier1");
    String valueStr = new String(cell.getValue());

    // 扫描特定列的所有数据
    Scanner scanner =
      table.getScanner(new String[]{"myColumnFamily:columnQualifier1"});
       RowResult rowResult = scanner.next();

    while(rowResult != null) {
```


互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
            System.out.println("Found row: " + new String(rowResult.getRow())
+ " with value: " +
             rowResult.get("myColumnFamily:columnQualifier1".getBytes()));

            rowResult = scanner.next();
        }

        // 另外一种方法是使用 foreach loop, Scanners 支持 iterable
        for (RowResult result : scanner) {
             System.out.println("Found row: " + new String(result.getRow())
+ " with value: " +
            result.get("myColumnFamily:columnQualifier1".getBytes()));
        }

        //完成的时候别忘了关闭
        scanner.close();
      }
    }
```

10.2 基于代理的分布式数据库

基于代理的分布式数据库的设计思想是在数据库和应用程序之间搭建一个中间层，对数据库访问进行代理。中间层屏蔽底层的数据库节点信息，甚至屏蔽关系数据库的类型（如底层可以是 Oracle、MySQL、PostgreSQL 等）。应用程序访问数据库时，就像直接访问单个数据库，基本上没有什么特殊区别。另外，由于中间层使用了连接池复用的技术，还可以避免由于数据库客户端过多导致的连接数超过数据库限制的问题。

这种数据库系统的框架如图 10-2 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

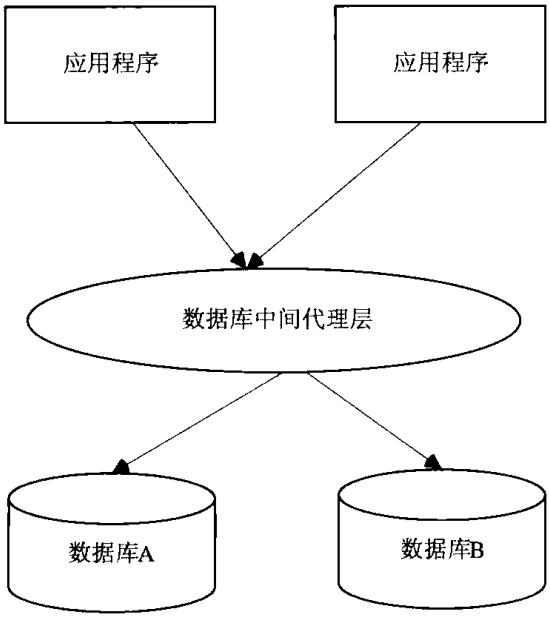


图 10-2　基于代理的分布式数据库框架

基于代理的分布式数据库主要解决以下问题：

（1）数据库连接数问题

由于目前的数据库提供网络服务时，都是基于每个连接一个进程或线程的方式，而且每个连接基本上要消耗 2MB 左右的内存。因此，当有很多数据库客户端访问数据库时，将会产生大量的连接，可能导致数据库主机的资源耗尽而停止提供服务。不过，实际上真正需要并行处理任务的连接，没有建立的连接多。因此，通过在中间代理层进行连接控制，使用共享连接池等技术可以解决连接过多导致数据库主机资源耗尽问题。

（2）读写分离

对于某些应用场景，写操作比较少，而读操作很频繁。因此，采用复制的方式，数据写到 Master 数据库，读操作分离到多个 Slave 数据库进行读操作的方式，可以提高数据库的性能。但直接采用读写分离，可能涉及应用程序的修改，应用程序访问数据库的时候需要区分是读操作还是写操作，增加了应用程序的复杂度。通过在中间代理层增加读写分离技术，这样对应用程序而言就是透明的，不用关心具体后台的读写分离操作，

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

对已有程序也不需要做迁移升级。代理层接收到应用程序的 SQL 语句调用时分析是读操作还是写操作，当是写操作，则分配到 Master 数据库上执行任务；若是读操作，则根据配置，当有多台 Slave 读数据库时，则可以使用平均公平等算法，把读操作分配到某台 Slave 读数据库上去执行任务，然后返回结果给应用程序。

（3）数据分库

对于一些场景，如某张表的数据量非常大，可能为几亿条记录，并且读写操作很频繁，达到每秒几千次写操作。这时，使用单一的数据库主机方式就很难达到性能要求。因此，通过按照行关键字对表进行拆分或许是一种选择。通过在代理层定义需要进行数据拆分的表和拆分规则，来屏蔽拆分方式对应用程序系统的改造，使之对应用开发人员透明。同时，对于汇总类的数据操作，可采用并行计算，即同时发送请求给所有后端被拆分的子数据库系统去执行任务，并把结果进行汇总后再返回给应用程序。一个应用程序的 SQL 写操作请求到达代理层时，代理层分析此 SQL 语句中的表是否定义了拆分规则。若有则取出拆分规则，得到相应的后端子数据库。把 SQL 语句发送到后端子数据库执行，并从后端子数据库得到结果返回给调用者。若此 SQL 语句是读操作，且没有指定规则中的行关键字字段，那么代理层将把此 SQL 语句发送给所有的后端子数据库（符合该表拆分规则的后端子数据库）进行并发执行，然后代理层把所有后端子数据库返回的结果进行汇总，然后再返回给调用者。

目前实现了以上几个功能的代理数据库，属于开源的数据库代理中间件有 PGPool-II。PGPool-II 是一个工作在 PostgreSQL 数据库服务器和 PostgreSQL 数据库客户端之间的中间件，它提供了以下几个功能：

（1）抑制连接限制过多

所有单个数据库都配置一个最大的连接数，当达到这个限制后，新的连接将被拒绝。若设置更大的连接数，将会消耗更多的数据库服务器的系统资源，并影响数据库的性能。因此，使用连接复用的连接池技术可以避免这个问题。PGPool-II 也有最大的连接限制，当超过最大值后，额外的连接会放到队列中而不是立即返回错误。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（2）连接池

PGPool-II 保存与数据库服务器的连接，放在连接池中。当数据库客户端进行新的连接时，若相同属性（数据库名称、用户名和密码、协议版本等）的连接已经存在于连接池中，则可以重用已有的连接，不需要与数据库建立新的连接，减少了连接建立的开销，增强了系统的吞吐量。

（3）复制

PGPool-II 能够管理多个 PostgreSQL 数据库服务器，使用复制技术，能够实时备份数据到多个磁盘上，避免了因为磁盘故障导致的停机服务。

（4）负载均衡

PGPool-II 利用复制技术，把查询请求分布到不同复制服务器上执行，降低了每台 PostgreSQL 数据库的压力，增加了系统的整体性能。在理想情况下，整体性能随着复制的数据库服务器增加而增加，尤其当同一时刻有许多查询执行的时候，负载均衡的功能最有用。

（5）并行查询

使用并行查询功能，数据能够分布到不同的服务器上，因此能够在所有服务器上执行并行查询，减少整体的执行时间。尤其当数据量非常大的时候，并行查询功能特别有用。

目前 PGPool-II 只支持 PostgreSQL 数据库。下面介绍配置和使用 PGPool-II 的几个操作模式和相关的一些重要参数说明。

PGPool-II 的配置文件默认是/etc/pgpool.conf 和/etc/pcp.conf。PGPool-II 支持多个操作模式，每一个模式有相关的功能可以 Enabled 或 Disabled，有特别的参数来控制它们的行为。表 10-6 说明了各个模式下的功能支持与否，“X”代表不支持，“O”代表支持。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 10-6　PGPool-II 功能模式对应表

功能/模式	裸模式	连接池模式	复制模式	主/备模式	并行查询模式
连接池	X	O	O	O	0
复制	X	X	O	X	(*)
负载均衡	X	X	O	O	(*)
失效切换	O	O	X	X	X
并行查询	X	X	X	X	O
需要服务器	1 或更多	1 或更多	2 或更多	2 或更多	2 或更多
是否需要系统 DB	不	不	不	不	是

（*）在并行查询模式下，对于启用分割数据的表，复制和负载均衡功能不能同时使用。

配置 pcp.conf 文件

PGPool-II 为系统管理员提供了通过网络来收集 PGPool-II 状态和终止 PGPool-II 的控制接口。pcp.conf 文件就是这类接口用于鉴权的用户/密码，所有的模式都需要 pcp.conf 文件。空行或以“#”开头代表注释。用户名和密码必须按照下面的格式写在同一行：

```
用户名：[MD5 过的密码]
```

[MD5 过的密码]由命令 pg_md5 生成，如：

```
./pg_md5 foo
acbd18db4cc2f85cedef654fccc4a4d8
```

下面简单介绍一下模式，以及模式中重要的几个相关参数信息。具体使用哪个模式是在配置文件 pgpool.conf 中进行配置的。

（1）裸（raw）模式：

在此模式下，客户端仅仅通过 PGPool-II 连接到 PostgreSQL 数据库，一般用于解决连接限制问题，以及数据库服务器的失效切换问题。下面介绍几个重要的参数，详细的参数请参考 PGPool-II 官方文档。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- **listen_addresses**：指定接受 TCP/IP 连接的服务器地址，可以是 hostname 或 IP 地址。“*”代表接受任何连接请求，默认是“localhost”，也就是说，连接是通过 UNIX domain socket 通信的。
- **Port**：指定接受连接的端口，默认是 9999。
- **backend_hostname**：用来指定后端数据库的地址，有多台时，是按照 backend_hostname 后面加数字的格式表示的，数字从 0 开始，如有 3 台，则为 backend_hostname0、backend_hostname1、backend_hostname2。
- **backend_port**：用来指定后端数据库的端口，有多台时，是按照 backend_port 后面加数字的格式表示的，数字从 0 开始，如有 3 台，则为 backend_port0、backend_port1、backend_port2。
- **backend_weight**：用于指定后端数据库的权重，默认为 1，即所有后端数据库都是平衡的。此字段与前面的 backend_port 一样，是配合使用的。

裸模式下的失效切换

在裸模式下，如果配置了多台后端数据库服务器，那么可以执行失效切换。在正常情况下，PGPool-II 通常是访问由 backend_hostname0 指定的数据库服务器。如果 backend_hostname0 失效，那么 PGPool-II 将会访问 backend_hostname1；如果 backend_hostname1 也失效，那么会访问 backend_hostname2、backend_hostname3。依此类推。

（2）连接池模式

- **max_pool**：用于指定在 PGPool-II 子进程中 Cache 的最大连接数，如果进来的连接使用相同的数据库名和用户名，则会重用 Cache 的连接，否则创建新的连接并放到 Cache 中。若超过最大的连接数，则把最旧的连接从 Cache 中移除。
- **connection_life_time**：用于指定 Cache 的连接过期时间，以秒为单位。过期的连接将被断开。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（3）复制模式

- **replication_mode**：这个参数设置为 true 时才启用复制模式，默认为 false。
- **replication_timeout**：指定复制的超时时间，用于检测死锁，单位是毫秒。
- **load_balance_mode**：当设置为 true 时，SELECT 查询将会分布到各个后端数据库查询，提高查询性能。

（4）并行模式

- **parallel_mode**：需要设置为 true。

另外，并行模式还需要配置 SystemDB，有几个相关的参数如下：

- **system_db_hostname**：此参数为存放 SystemDB 的数据库服务器的地址。
- **system_db_port**：此参数用于存放 SystemDB 的数据库服务器的端口。
- **system_dbname**：此参数用于存放 SystemDB 的数据库服务器的数据库名称。
- **system_db_schema**：此参数用于存放 SystemDB 的数据库服务器的模式。
- **system_db_user** ：此参数用于访问存放 SystemDB 的数据库服务器的用户名。
- **system_db_password**：此参数用于访问存放 SystemDB 的数据库服务器的密码。

使用并行模式，是有一些限制的，不是所有操作都能正常使用。

并行模式下的一些限制如下：

- INSERT

不能在分区字段中使用 DEFAULT。比如，如果 x 是数据表 t 的分区字段，则下边的语句无效：

```
INSERT INTO t(x) VALUES (DEFAULT);
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

同样，函数也不能作为值使用：

```
INSERT INTO t(x) VALUES (func());
```

在这里必须使用常量。

另外，也不能支持 SELECT INTO 和 INSERT INTO ... SELECT 语句。

- UPDATE

如果更新分区列可能会丧失数据一致性，PGPool-II 不能对更新过的数据重新分区。

如果查询因为违反约束（constraint）而引发错误，那么事务不能回滚（rollback）。

如果从 WHERE 子句中调用函数，更新可能无法正确执行。比如：

```
UPDATE branches set bid = 100 where bid = (select max(bid) from beances);
```

- SELECT ... FOR UPDATE

如果从 WHERE 子句中调用函数，查询可能无法正确执行。比如：

```
SELECT * FROM branches where bid = (select max(bid) from beances) FOR
UPDATE;
```

- COPY

不支持 COPY BINARY，也不支持从文件中 COPY，只能支持 COPY FROM STDIN 和 COPY TO STDOUT。

- ALTER/CREATE TABLE

更新分区规则（partition rule）以后，必须重新启动 PGPool-II 从系统 DB 中读入新设置。

- Transaction

在事务块中执行的 SELECT 语句将被放在分离的事务块中执行。例如：

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
BEGIN;
INSERT INTO t(a) VALUES (1);
SELECT * FROM t ORDER BY a; <-- 上边的 INSERT 结果在这个 SELECT 语句中不可见 END;
```

- Views / Rules

对于视图和规则，后端数据库将创建相同的定义。

```
SELECT * FROM a, b where a.i = b.i
```

像上边这样的联合（join）将在每一个后端数据库运行，然后合并它们的结果，无法建立跨越节点的视图和规则。

（看起来这里有个问题，如果存在分区，那么 a 和 b 必须基于 i 而且必须一样。）

- Functions / Triggers

相同的函数定义在每一个后端数据库创建，跨越节点的联合（join）和其他节点上的数据无法在函数内部操作。

- Extended Query Protocol

不支持被 JDBC 驱动等程序使用的扩展查询协议（extended query protocol），必须使用简单查询协议（simple query protocols）。

- Multi-byte Characters

PGPool-II 不能进行不同字符集之间的转换，因此客户端、后端数据库和 PGPool-II 系统 DB 的字符集编码必须相同。

- Deadlocks

跨越后端服务器的死锁无法被检测。例如：

（tellers 数据表按照以下规则分区-partition）

```
tid <= 10 --> node 0
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
tid >= 10 --> node 1
 A) BEGIN;
 B) BEGIN;
 A) SELECT * FROM tellers WHERE tid = 11 FOR UPDATE;
 B) SELECT * FROM tellers WHERE tid = 1 FOR UPDATE;
 A) SELECT * FROM tellers WHERE tid = 1 FOR UPDATE;
 B) SELECT * FROM tellers WHERE tid = 11 FOR UPDATE;
```

这个例子，单一节点无法检测死锁，PGPool-II 会一直等待响应，这个现象发生在其他检索获得行级锁（row level lock）时。可以设置 replication_timeout 参数避免这个问题。

同样，如果某个节点发生死锁，每个节点的事务状态会变得不一致，所以，如果检测到死锁，PGPool-II 会中止处理。

```
pool_read_kind: kind does not match between master(84) slot[1] (69)
```

- Schemas

public 之外的 schema 的对象必须用全名引用：

```
schema.object
```

否则，当设置路径 set search_path = xxx 并且在查询中省略 schema 名时，PGPool-II 将无法解析正确的 schema。

接下来举两个例子说明如何配置和使用。

准备数据库环境：从 postgreSQL 网站下载最新版本，目前是 8.3.3，可以使用预编译好的包。我们使用 RedHat 的 RPM 包，可以从 http://yum.pgsqlrpms.org/8.3/redhat/rhel-4AS-i386/上下载，分别下载：

http://yum.pgsqlrpms.org/8.3/redhat/rhel-4AS-i386/postgresql-libs-8.3.3-1PGDG.rhel4.i386.rpm

http://yum.pgsqlrpms.org/8.3/redhat/rhel-4AS-i386/postgresql-8.3.3-1PGDG.rhel4.i386.rpm

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

http://yum.pgsqlrpms.org/8.3/redhat/rhel-4AS-i386/postgresql-server-8.3.3-1PGDG.rhel4.i386.rpm

http://yum.pgsqlrpms.org/8.3/redhat/rhel-4AS-i386/pgpool-II-2.1-1.rhel4.i386.rpm

进行安装：rpm –i postgresql*.rpm rpm –i pgpool-II-2.1-1.rhel4.i386.rpm。

安装好后，配置 PG 数据库服务器，我们假设有 3 个后台 PostgreSQL 数据库服务器，在本地机器上运行，端口分别为 5431、5432、5433。

安装好后，默认的 postgres 用户被创建，其 home 目录在/var/lib/pgsql 下。

```
su - postgres #/var/lib/pgsql
```

开始初始化数据库环境：使用 initdb，该命令会在/var/lib/pgsql/data 目录下建立数据库服务器所需的环境文件。

然后我们在/var/lib/pgsql 目录下建立 3 个目录，分别为 pg1、pg2、pg3。

接下来拷贝先前初始化的数据库文件，使用以下命令：

```
cp -r /var/lib/pgsql/data /var/lib/pgsql/pg1/
cp -r /var/lib/pgsql/data /var/lib/pgsql/pg2/
cp -r /var/lib/pgsql/data /var/lib/pgsql/pg3/
```

修改/var/lib/pgsql/pg1/data 下的 postgresql.conf 文件的下面几个参数：

```
data_directory = '/var/lib/pgsql/pg1/data'
hba_file = '/var/lib/pgsql/pg1/data/pg_hba.conf'
ident_file = '/var/lib/pgsql/pg1/data/pg_ident.conf'
```

接着修改/var/lib/pgsql/pg2（3）/data 下的 postgresql.conf，按照上面的方法。

这样 3 个数据库实例在本地机器上已经配置好了，可以启动 3 个数据库实例了，启动方法如下：

```
pg_ctl -D /var/lib/pgsql/pg1/data start
pg_ctl -D /var/lib/pgsql/pg2/data start
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
pg_ctl -D /var/lib/pgsql/pg2/data start
```

现在可以看到 3 个数据库服务器实例已经启动，可以使用 psql 命令登录查看：

```
-bash-3.00$ psql -p 5431
postgres=# select datname from pg_database;
datname
-----------------
template1
template0
postgres
(3 rows)
```

可以看到有 3 个数据库默认存在。到这里为止，数据库后台已经配置完成，准备工作完成，接下来要使用 PGPool-II 的两个模式了。

1. 使用复制模式

PGPool-II 使用两个配置文件：pgpool.conf 和 pcp.conf，前面已经介绍过。编辑 /etc/pgpool.conf 文件，加入以下参数：

```
listen_addresses = '*'
port = 9999
pcp_port = 9898
replication_mode = true  #打开复制模式
load_balance_mode = true  #启用负载均衡模式
#第一个后台数据库
backend_hostname0 = '127.0.0.1'
backend_port0 = 5431
backend_weight0 = 1
#第二个后台数据库
backend_hostname1 = '127.0.0.1'
backend_port1 = 5432
backend_weight1 = 1
#第三个后台数据库
backend_hostname2 = '127.0.0.1'
```


互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
backend_port2 = 5433
backend_weight2 = 1
```

配置 PCP 命令配置文件，编辑/etc/pcp.conf 文件：

```
postgres:098f6bcd4621d373cade4e832627b4f6
```

意思是使用 postgres 账户，密码是 test，文件里存放的是 pg_md5 生成的 MD5 密码，由下面方法生成：

```
pg_md5 test
098f6bcd4621d373cade4e832627b4f6
```

现在复制模式配置已经完成，可以启动 PGPool-II 了，启动方式如下：

```
pgpool
```

停止方式为：pgpool stop，若要强制停止，方式为：pgpool -m fast stop。

现在我们检查复制模式是否配置成功?

首先我们创建一个需要复制的数据库，使用 createdb 创建，这个数据库需要在所有 3 个数据库后台被创建，创建方式如下：

```
createdb -p 9999 test_replication
```

然后我们使用 psql 命令查看是否在 3 个后台数据库服务器上都创建了。

```
-bash-3.00$ psql -p 5432
postgres=# select datname from pg_database;
    datname
-----------------
template1
template0
postgres
test_replication
(4 rows)
```

可以看到多出了一个数据库，说明 createdb 已经被 PGPool-II 执行成功，而且在后

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

台的 3 个数据库服务器上自动复制出来了。

接着我们使用 pgbench 命令，使用-i 选项来初始化数据库表和数据。

```
pgbench -i -p 9999 test_replication
```

这条命令理论上应该会在后台 3 个数据库服务器上创建以下表和数据，若 3 个数据库服务器上都有相同表和数据，则说明复制模式工作成功。

表　名	记录行数
branches	1
tellers	10
accounts	100000
history	0

我们使用 psql –p 5431, psql –p 5432、psql –p 5433 分别检查，命令如下：

```
test_replication=# \d
        List of relations
 Schema |   Name   | Type  |  Owner
--------+----------+-------+----------
 public | accounts | table | postgres
 public | branches | table | postgres
 public | history  | table | postgres
 public | tellers  | table | postgres
(4 rows)
test_replication=# select count(*) from tellers;
 count
-------
    10
(1 row)

test_replication=# select count(*) from history;
 count
-------
```


互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
     0
(1 row)

test_replication=# select count(*) from branches;
 count
-------
     1
(1 row)

test_replication=# select count(*) from accounts;
 count
--------
 100000
(1 row)
```

从结果中可以看出，所有表和数据在 3 个数据库实例上都被创建了。

2. 配置并行查询模式

在并行查询模式里，可以使用分区技术把不同范围的数据存储到不同的后端数据库服务器节点中。在 PGPool-II 里，使用并行模式，必须设置另外一个叫做“系统数据库”（System Database）的数据库，简称为 SystemDB。SystemDB 存放了用户定义的需要分区的数据表，以及分区规则。SystemDB 另外的用途是对使用 dblink 从后台数据库服务器发送回来的数据进行合并。这里我们继续使用前面配置的 3 个数据库节点。

（1）配置并行模式

为了启用并行模式，设置 parallel_mode 为 true：

```
parallel_mode=true
replication_mode=false
load_balance_mode = true
```

（2）配置 SystemDB

SystemDB 和普通的数据库没有什么特殊区别，只是它是用于存放需要分区的表和

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

分区规则，以及存放 dblink 使用的函数。我们在 5431 端口对应的数据库节点中创建 SystemDB，因此在 pgpoo.conf 中设置相关参数：

```
system_db_hostname = '127.0.0.1'
system_db_port = 5431
system_db_dbname = 'pgpool'
system_db_schema = 'pgpool_catalog'
system_db_user = 'pgpool'
system_db_password = ''
```

在 5431 中创建 SystemDB：

```
-bash-3.00$ createuser -p 5431 pgpool
-bash-3.00$ createdb -p 5431 -O pgpool pgpool
```

（3）安装并配置 dblink

前提是安装了 contrib rpm 包，安装好后默认的文件目录是/usr/share/pgsql/contrib/dblink.sql：

```
-bash-3.00$ psql -f /usr/share/pgsql/contrib/dblink.sql -p 5431 pgpool
```

初始化 SystemDB 表结构：

```
-bash-3.00$ psql -f /usr/share/pgpool-II/system_db.sql -p 5431 -U pgpool
pgpool
```

system_db.sql 文件是 PGPool-II 自带的，可以通过以下命令找到：

```
# rpm -ql pgpool-II|grep system
/usr/bin/pcp_systemdb_info
/usr/share/pgpool-II/system_db.sql
```

System_db.sql 文件里的内容主要是：

```
CREATE SCHEMA pgpool_catalog;
```

定义分区规则表：

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
CREATE TABLE pgpool_catalog.dist_def(
        dbname TEXT,
        schema_name TEXT,
        table_name TEXT,
        col_name TEXT NOT NULL CHECK (col_name = ANY (col_list)),
        col_list TEXT[] NOT NULL,
        type_list TEXT[] NOT NULL,
        dist_def_func TEXT NOT NULL,
        PRIMARY KEY (dbname,schema_name,table_name)
);
```

定义复制规则表：

```
CREATE TABLE pgpool_catalog.replicate_def(
        dbname TEXT,
        schema_name TEXT,
        table_name TEXT,
        col_list TEXT[] NOT NULL,
        type_list TEXT[] NOT NULL,
        PRIMARY KEY (dbname,schema_name,table_name)
);
```

定义查询 Cache 规则表：

```
CREATE TABLE pgpool_catalog.query_cache (
        hash TEXT,
        query TEXT,
        value bytea,
        dbname TEXT,
        create_time TIMESTAMP WITH TIME ZONE,
        PRIMARY KEY(hash, dbname)
);
```

接下来就是定义分区规则了。首先定义我们要对哪张表进行分区，通过在 dist_def 中插入数据实现，例如：

```
INSERT INTO pgpool_catalog.dist_def VALUES (
```


Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
    'bench_parallel',
    'public',
    'accounts',
    'aid',
    ARRAY['aid', 'bid', 'abalance', 'filler'],
    ARRAY['integer', 'integer', 'integer', 'character(84)'],
    'pgpool_catalog.dist_def_accounts'
);
```

上面的 SQL 语句的意思是对 bench_parallel 数据库名下 public 模式下的 accounts 表数据进行分区，分区是按照 aid 字段进行的，分区规则函数是 dist_def_accounts，该函数定义如下：

```
CREATE OR REPLACE FUNCTION pgpool_catalog. dist_def_accounts (anyelement)
RETURNS integer AS $$
    SELECT CASE WHEN $1 > 0 AND $1 <= 100000THEN 0
        WHEN $1 > 100000 AND $1 <= 200000 THEN 1
        ELSE 2
    END;
$$ LANGUAGE sql;
```

此函数的意思是如果 aid 大于 0 且小于等于 100000，则该行数据存放到 5431 对应的数据库中；若 aid 大于 100000 且小于等于 200000，则该行数据存放到 5432 的数据库中；其他值则存放到 5433 对应的数据库中。

定义复制规则：只对特定表启用复制功能，我们这里对 branches 和 tellers 表启用复制功能：

```
INSERT INTO pgpool_catalog.replicate_def VALUES (
    'bench_parallel',
    'public',
    'branches',
    ARRAY['bid', 'bbalance', 'filler'],
    ARRAY['integer', 'integer', 'character(88)']
);
```


互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
INSERT INTO pgpool_catalog.replicate_def VALUES (
    'bench_parallel',
    'public',
    'tellers',
    ARRAY['tid', 'bid', 'tbalance', 'filler'],
    ARRAY['integer', 'integer', 'integer', 'character(84)']
);
```

然后重启 PGPool-II，确定 pgpool.conf 文件参数如下：

```
listen_addresses = '*'
port = 9999
pcp_port = 9898
parallel_mode=true
replication_mode =false
load_balance_mode = true

system_db_hostname = '127.0.0.1'
system_db_port = 5431
system_db_dbname = 'pgpool'
system_db_schema = 'pgpool_catalog'
system_db_user = 'pgpool'
system_db_password = ''

backend_hostname0 = '127.0.0.1'
backend_port0 = 5431
backend_weight0 = 1
backend_hostname1 = '127.0.0.1'
backend_port1 = 5432
backend_weight1 = 1
backend_hostname2 = '127.0.0.1'
backend_port2 = 5433
backend_weight2 = 1
```

（4）检查并行查询模式的有效性

首先创建一个需要分区的数据库，这个数据库需要在所有的后端数据库中创建。通

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

过 PGPool-II 将在所有的后端数据库中创建，使用命令如下：

```
-bash-3.00$ createdb -p 9999 bench_parallel
```

然后使用 pgbench 来初始化表结构和数据：

```
-bash-3.00$ pgbench -i -s 3 -p 9999 bench_parallel
```

接着查看 3 个数据库节点的数据，可以看到每个数据库的数据行数各为 100000。可以使用下面命令检查是否正确：

```
-bash-3.00$ psql -p 5431 bench_parallel "select count(*) from accounts;"
count
--------
 100000
(1 row)
-bash-3.00$ psql -p 5432 bench_parallel "select count(*) from accounts;"
count
--------
 100000
(1 row)
-bash-3.00$ psql -p 5432 bench_parallel "select count(*) from accounts;"
count
--------
 100000
(1 row)
```

从结果中可以看出，每个数据库节点的数据是 100000，刚好平均分配，说明数据分区成功。

再从 PGPool-II 看总数是否正确：

```
-bash-3.00$ psql -p 9999 bench_parallel -c "select count(*) from
accounts;"
 count
--------
 300000
(1 row)
```


互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

从上面结果中可以看出，PGPool-II 确实执行了并行查询和数据分区功能。

基于代理的分布式数据库的缺陷如下：

- 基于代理的分布式数据库的分区解决了部分问题，但可扩展性不够强，当整个系统配置好后，就无法再扩容了。因为数据是按照列进行分区的，后端数据库节点不能在线增加，新的数据无法分布到新的数据库节点上。
- 由于数据库客户端所有的执行操作都要经过中间层，单个服务器也将成为瓶颈。有测试表明，当后端数据库服务器超过 6 个的时候，性能不再随着后端数据库的增加而增加，反而开始下降。
- 对于单个代理服务器本身的性能瓶颈，也有办法解决，就是在代理服务器前面使用 VIP 方式。一个 VIP 对应多个代理数据库服务器，数据库客户端通过 VIP 连接代理服务器，这样就可以有多个代理服务器提供服务。但还是无法解决后端数据库的扩容问题。

10.3 基于列的分布式关系数据库

在前面基于多租户的章节中，已经提到了基于多租户的分布式数据库，其实该方案可以归纳抽象成一个共性的分布式数据库系统。

为了解决数据库系统的水平扩展和在线扩容以及更高性能，我们设计了一个叫做 MapDDB 的按行关键字分布的分布式数据库系统。该系统已经在阿里软件的阿里旺旺系统中使用，效果很好，性能非常高。它的主要设计思路是：数据库中的行数据是有共同特性的，可以按照某个列（行关键字）进行分割。这个列不能为空，一般是一行数据中的主键，按照列把数据分割到不同的后端数据库中。数据库应用程序使用 JDBC/ODBC/C++接口访问数据库，有一个列管理系统，负责列和后端数据库的对应关系。应用程序访问实际的后端数据库之前先查询该列字段（行关键字）对应的后端数据库，然后得到后端数据库的连接对象后直接访问后端的实际数据库进行数据的存取。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

该系统具有以下特点：

- 支持按照行关键字对表数据进行分割，相同行关键字的行数据在同一个子数据库的表里；
- 支持在线增加子数据库节点，扩容时，不需要停机即可把新的数据分布到新的子数据库节点；
- 使用普通 PC 服务器存储数据；每个数据库服务器节点配置支持：
 - ✧ 多种类型关系数据库，目前支持 Oracle、MySQL、PostgreSQL；
 - ✧ 两种语言接口开发应用，支持 C++和 Java 接口；
 - ✧ 在线进行数据在不同子数据库之间的迁移。

MapDDB 系统的大体设计框架如图 10-3 所示。

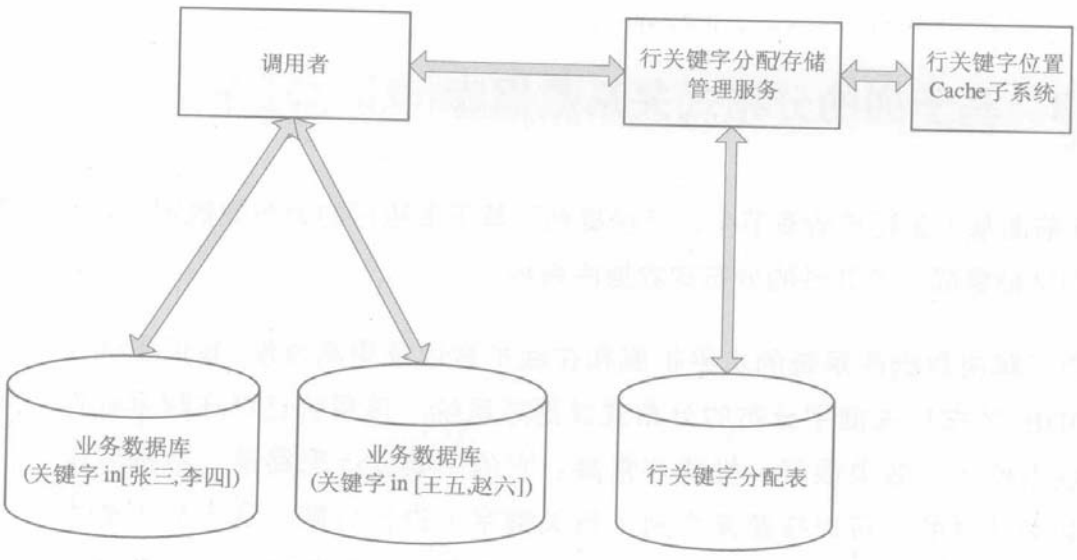


图 10-3　MapDDB 系统架构

目前该系统支持的后端数据库包括：Oracle、MySQL、PostgreSQL 等，理论上只要支持 JDBC 接口的关系数据库都可以简单地集成支持。为了测试这套分布式系统的性能，我们搭建了一个性能测试环境进行测试，如下：

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

我们准备了 1200 万行的数据文件，数据是用户的登录/登出日志，大概是 100 万用户。

我们搭建了一个由 7 个子数据库节点组成的分布式数据库系统，子数据库使用 MySQL 5.0 版本，每个子数据库节点有一台对应的服务器对数据进行复制，使用 InnoDB 数据引擎，打开 log-bin 复制功能，以及 syncbin=5，相当于 14 台子数据库服务器，其中 7 台作为主提供数据服务，另外 7 台作为备提供数据服务。每台数据库服务器的配置都一样，使用 PC 服务器，配置 16GB 内存，2 个 4 核的 AMD CPU，16 块 SAS 硬盘，每块硬盘 146GB 空间。其中 2 块用于操作系统本身，操作系统使用 RedHat 企业板 4.x64 位版本，14 块硬盘组成 RAID5 作为数据库的数据存储。这样大概每台服务器提供 1.6TB，7 组总共提供 13TB 的数据空间。

这里特征字使用用户账户。用户账户与数据库的对应关系的中央 Cache 也使用 MySQL 来做持久化。由于在现实情况中，新用户的增加是非常少的，平均每秒不超过 20 个。因此为了避免用户与数据库对应关系的持久化带来的开销，在测试前先把用户和数据库的对应关系按正常情况进行分配好。

使用 2 台服务器搭建内存 Cache 系统，使用 2 台服务器搭建 MapDDB，使用 VIP 对外提供统一服务。测试程序使用 C++开发，读取数据文件，使用 60 个线程分割，同时向分布式数据库系统中插入数据。子数据库配置测试分为 4 个用例：

只配一个子数据库节点；配置 2 个子数据库节点；配置 4 个子数据库节点；配置 7 个数据库节点。这样 4 个用例用来证明是否随着子数据库节点的增加，数据的插入性能是线性增加的，即分布式系统的容量和性能可以通过在线增加子数据库服务器的方式来达到。

然后分别在 4 种配置情况下对 1200 万记录进行插入操作、删除操作、查询操作和更新操作的每秒可以执行的次数

如表 10-7 和图 10-4 所示是测试结果。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

表 10-7　MapDDB 性能测试结果

测试方法	子数据库服务器的节点数目			
	1	2	4	7
插入	1800	3560	7110	12550
查询	9200	18000	35800	63400
删除	3747	7420	14021	24580
更新	2306	4524	8960	15204

单位是每秒的次数

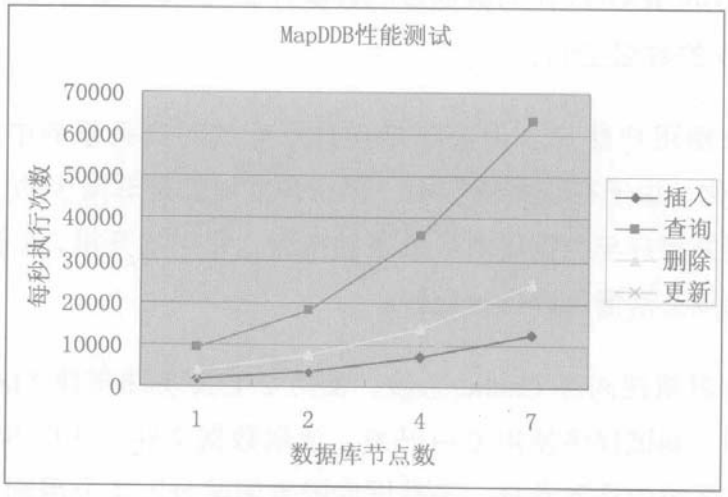


图 10-4　MapDDB 性能测试结果图

从测试结果可以看出，设计符合要求，并且性能随数据库节点增加而增加，支持线性扩容。

若某个行关键字拥有的数据非常多，而且读写非常频繁，那么一台数据库服务器可能无法承受压力。在这种情况下，可以对 MapDDB 做适当扩充。MapDDB 中的子数据库可以换成前面介绍的基于代理的分布式数据库系统，而基于代理的分布式数据库可以实现读写分离，以及并行查询或者表数据的水平分割。把属于同一行关键字的一张表再按照表中的某个字段进行水平分割到不同的后端数据库中，这相当与进行两级水平分割。基于代理的 MapDDB 系统架构如图 10-5 所示。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

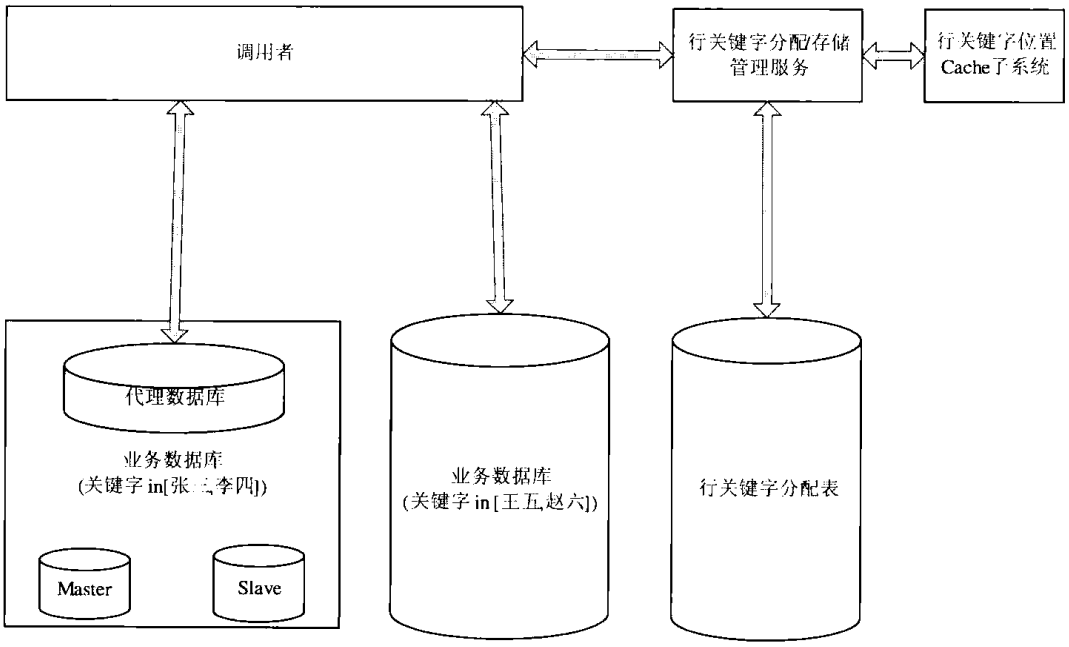


图 10-5　基于代理的 MapDDB 系统架构

10.4　小结

本章介绍了 3 种分布式数据库模型，适用于不同的应用场景：

- 如对于非关系型业务应用，若数据量很大，而且是稀疏型表结构，那么使用基于列的结构化分布式数据库比较合适；
- 若数据量一般，而且总的数据量可以估计，那么可以使用基于代理的分布式数据库来解决数据库压力问题；
- 若数据量很大，而且需要使用传统的关系型数据库功能，那么基于列的分布式关系数据库比较合适。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第11章 分布式Cache

11.1 单层的分布式 Cache ……243
11.2 多层的分布式 Cache ……245
11.3 改进的多层分布式 Cache ……245
11.4 小结 ……246

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在前面章节的 ADFS 中，由于文件数是海量的，因此对于海量的文件数目中实现一个快速的文件 Meta（文件所在的块服务器和副本信息）信息，就尤其重要。为了实现快速查询，我们使用了一套分布式 Cache 系统，来加速文件的快速定位。分布式 Cache 系统使用支持Consistent Hashing算法的MemcachedClient库和开源的MemcachedServer。下面介绍一下 Consistent Hashing 算法和 Memcached 服务。

11.1 单层的分布式 Cache

Memcached 是 danga.com 的一个项目，最早是为 LiveJournal 服务的。目前全世界不少人使用这个缓存项目来构建自己的大负载网站，来分担数据库的压力。它的主要思想是使用 Key 和 Value 的简单数据格式，把 Key 和 Value 的对应关系使用 Hash 表的形式存放在内存中，并使用 LRU 算法把过期的数据从内存中移出，或则当内存占用超过阀值时，也是采用 LRU 算法把最近不用的数据从内存中移出，以保证程序不会因为内存用完而死掉。它保证了最近频繁访问的数据存放在内存中，可以极大地降低数据库或后台服务器的压力。Memcached 主要提供了 get(s)/set(s)接口，支持对数据缓存的存取。

Consistent Hashing 的基本算法如下：

整个系统里使用相同的 Hash 算法。首先设定一个 2^n 的值，这里假定 n=32，那么是 4294967296。然后对每个 Cache 节点进行 Hash 计算（如使用提供服务的 IP 和端口进行 Hash 计算），得到每个 Cache 节点的一个 Hash 值，把这个 Hash 值放在这个 DHT（Distributed Hash Tables）圆里。若有 N 个节点，则整个圆被分割成 N 部分，每个 Key 都是分布在这个圆上的一个点。我们假设要把 K_i 存放到分布式 Cache 系统中，首先计算 K_i 的 Hash 值得到 V_i，然后把 K_i 对应的 Hash 值 V_i 也放在圆里。从 V_i 的位置开始顺时针找到第一个可用的服务节点 N_i，然后把 K_i 保存到 N_i 节点上，也就是说，Hash 值在 N_x 和 N_y（节点的 Hash 值，N_x 和 N_y 在圆上是顺时针）之间的所有 K 都保存到 N_y 节点上。这种算法的好处是可以动态增加或减少内存 Cache 节点的数目，增加/减少一个节点，只会影响 K/N 个数据（K 为节点上的数据总数），每个原有 Cache 节点的$((N-1)/N)*100\%$的数据 Cache 仍旧有效。有些增强型还支持虚拟节点，用来更加平衡化每个 Cache 服务器

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

的 Key 数目，因为使用一般的 Hash 函数的话，映射 Key 到节点的分布会非常不平均。因此使用虚拟节点的思想，为每个物理 Cache 节点在圆上分配 100～200 个虚拟点，这样可以抑制 Key 的分布不均匀，最大限度地减少 Cache 服务器增减时的 Cache 重新分布。

Consistent Hashing 圆环如图 11-1 所示。

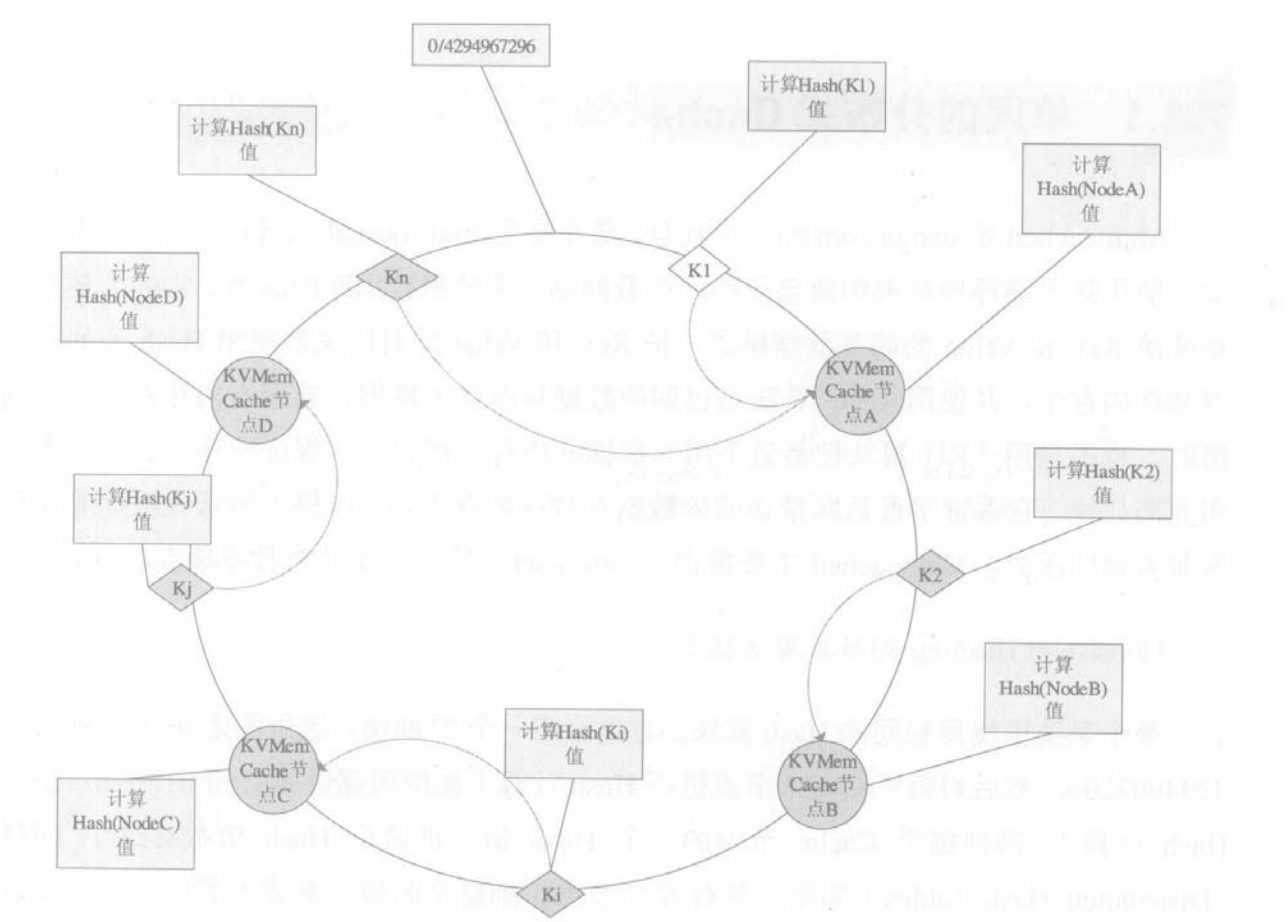


图 11-1　Consistent Hashing

目前支持 Consistent Hashing 算法的库有：

- libketama，下载地址是 http：//static.last.fm/ketama/ketama-0.1.1.tar.bz2；
- libmemcached，下载地址是 http：//tangent.org/552/libmemcached.html；
- Java API for memcached，下载地址是 http：//www.whalin.com/memcached/。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

11.2 多层的分布式 Cache

使用单层的分布式 Cache 基本上就可以很好地满足绝大部分 Cache 功能需求，但对于非常频繁进行查询的系统来讲，还是有点缺陷。主要原因是需要频繁进行远程调用，因此远程调用会成为一个潜在的瓶颈。为了解决这个问题，我们在上面的 Cache 系统基础上引入了多层的概念，即 Cache 的数据进行容余。在调用者本地也存放 Cache，这样可以减少远程调用的次数，也就提高了整体的性能。系统架构图如图 11-2 所示。

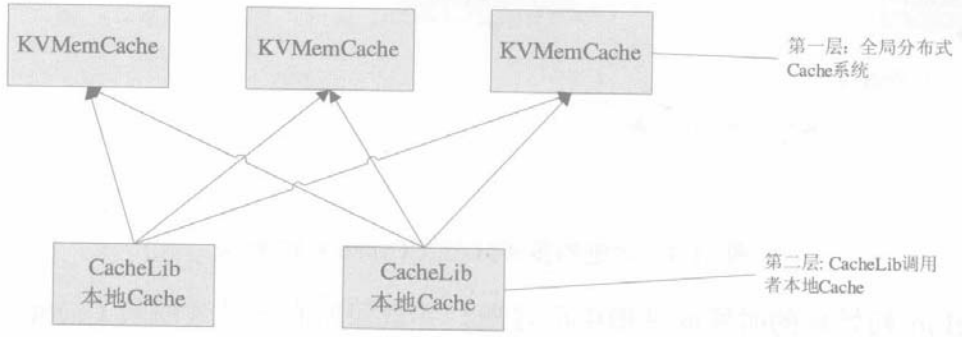


图 11-2 组播方式的多层分布式 Cache 系统架构

这种方式也带来了缺点，即 Cache 更新同步问题。因为，每个 CacheLib 本地 Cache 中的数据包含各个 Memcached 服务器中各个 Key 的数据，而某些 key 相关的数据可能会被更改，因此会导致 Cache 数据的不一致。解决的办法是增加 Cache 同步功能。简单的解决办法是使用 UDP 组播方式，当 Key 相关的数据被更改时，发出组播命令告诉其他的应用方 CacheLib，本地 Cache 里的 Key 无效或者进行更新。这种方式的缺点也是明显的，就是网络环境比较复杂，即所有的使用 CacheLib 本地 Cache 功能的应用程序必须处于同一组播段。另外，由于组播是 UDP，是不可靠通信，无法确保所有的应用程序都接收到了组播命令。

11.3 改进的多层分布式 Cache

如何解决组播引起的不可靠性带来的Cache不一致性问题？我们需要对上面的架构

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

图做些改进，新的架构图如图 11-3 所示。

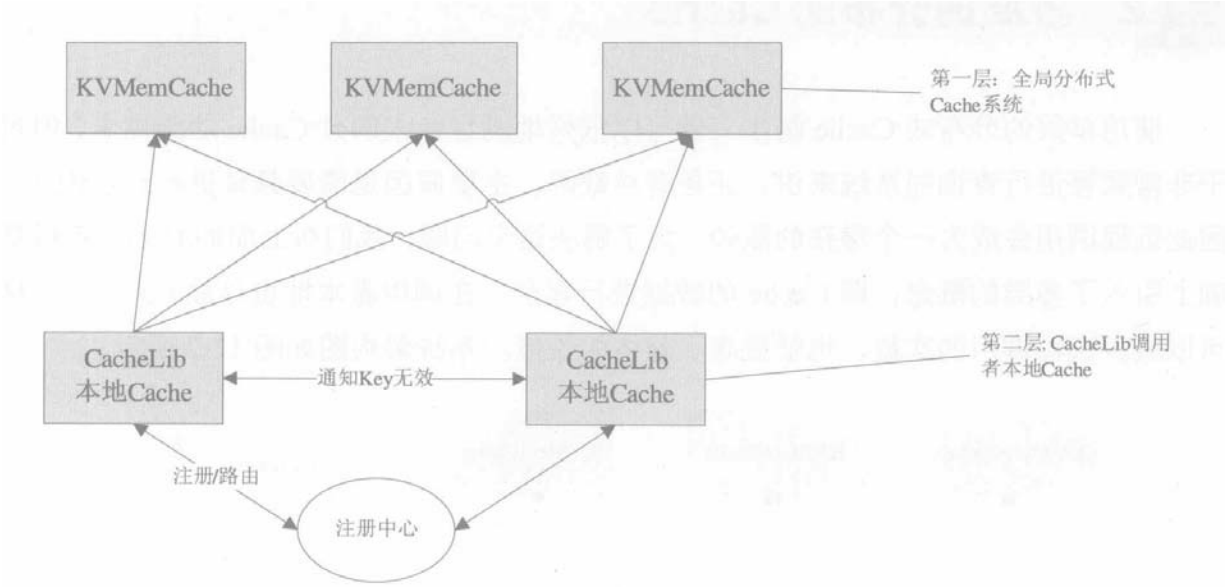


图 11-3　改进的多层分布式 Cache 系统架构

CacheLib 初始化的时候向注册中心注册，并取回所有已经注册的 CacheLib 的开放的 IP/Port，然后所有的 CacheLib 之间建立全网连接。当有 Key 的数据进行更改操作时，首先通过 TCP，向所有建立连接的其他的 CacheLib 进行 Key 无效通知，然后再更改 Key 的数据到 Key 所在的 Memcached 服务器，并更新本地 Cache。因为使用 Cache 的前提是 Key 的数据以读取为主，更新很少，因此使用全网通知的网络开销不会很大。这样采用两阶段的方式可以明显地避免 Cache 的不一致性问题。

这种方式的网络部署方便，而且可靠性更高。阿里软件的阿里旺旺即时通信系统就使用了多层的分布式 Cache 系统来解决大并发在线问题。

11.4　小结

本章介绍了 3 种不同的分布式 Cache 方案。根据企业的数据情况和存取情况不同，可以采用不同的方式：

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 若数据量大，但读取频率不高，如小于 10000 次/秒，则可以使用单层的分布式 Cache 系统；
- 对于数据量很大，读取频率又非常高，如大于 50000 次/秒的情况，则建议使用多层的分布式 Cache 系统；
- 像即时系统这种对即时性要求非常高，数据量又很大的情况，使用改进的多层分布式 Cache 系统比较合适。

SaaS 架构设计

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第12章 分布式计算

12.1 MapReduce 分布式计算框架 ……249
12.2 文件下载的分布式计算 ……252
12.2.1 中心索引方式 ……253
12.2.2 分布式的结构化方式 ……254
12.2.3 改进模型 ……256
12.3 小结 ……258

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在现实环境中，除了需要解决海量的存储问题之外，还要解决海量的数据计算、文件下载更新等问题。而解决这些问题的一个好的方法是使用分布式计算，下面主要对两个方面的现实问题进行简单介绍。

- 一是基于 MapReduce 的分布式计算；
- 二是文件下载的分布式下载方式。

12.1 MapReduce 分布式计算框架

在 CRM 这个例子中，需要分析用户的操作行为，以及分析业务组件的使用情况等，以了解用户最关心的功能和优化系统结构。这一般都需要通过分析日志来得出最终结果。由于日志文件非常大，而且多，普通的分析计算非常慢。因此，就需要新的分布式计算框架来解决日志分析问题。由于日志文件已经存在于前面的分布式存储系统中了，刚好可以利用该分布式存储来搭建分布式计算。MapReduce 分布式计算框架刚好是该类问题的最好解决方案。

Google 的 MapReduce 论文正式提出了 MapReduce 的分布式计算框架。MapReduce 的计算核心思想就是把数据和任务进行分解，让不同的计算机同时去执行部分数据和任务，这个步骤叫做 Map；然后产生中间结果集，并把所有中间结果集进行汇总，产生最终的结果集返回给计算调用者，这个步骤叫做 Reduce。对于详细的 MapReduce 计算的理论，请参考 Google 的论文 http://research.google.com/archive/mapreduce.html。Hadoop 不仅提供了分布式文件的存取，也提供了基于分布式文件的 MapReduce 分布式计算框架。用户可以使用 Hadoop 进行基于 MapRedeuce 方式的分布式应用计算。Hadoop 的设计是参考 Google 的 MapReduce 的，因此理解了 Hadoop 的设计、使用和开发，也就基本上理解了 MapReduce 这种分布式计算技术。相关的实践方式，可以参考 Hadoop，进行具体开发实践。

Hadoop 实现的 MapReduce 的技术框图如图 12-1 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

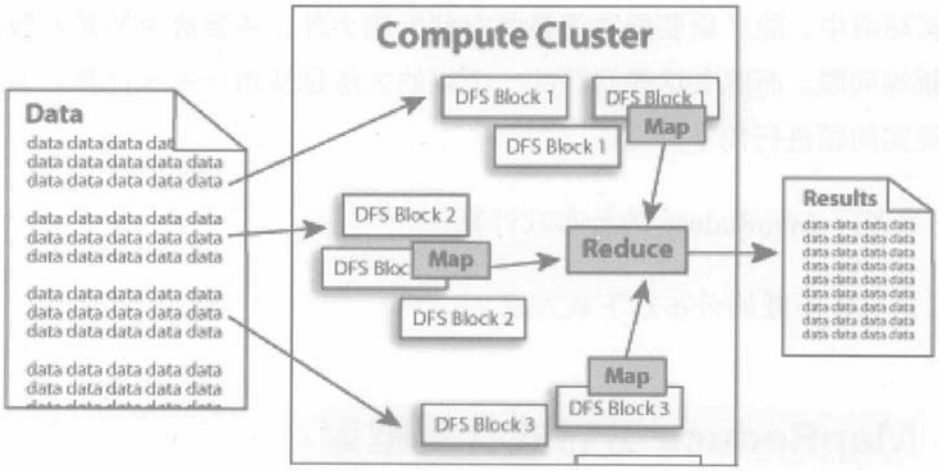


图 12-1　Hadoop 实现的 MapReduce 的技术框图

■ 基于 Hadoop 进行 MapReduce 开发应用

比如下面的例子，可以计算在 Hadoop 文件系统中存储的一个大文件（10GB）的某个词（“奥运会”）出现的次数。

源代码如下：

```
package org.myorg;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
public class WordCount {
   public static class Map extends MapReduceBase implements Mapper
<LongWritable, Text, Text, IntWritable> {
     private final static IntWritable one = new IntWritable(1);
     private final static String findWord= “奥运会”
     private Text word = new Text();
     public void map(LongWritable key, Text value, OutputCollector<Text,
```


互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
IntWritable> output, Reporter reporter) throws IOException {
          String line = value.toString();
          //StringTokenizer tokenizer = new StringTokenizer(line);
          int fromIndex=0;
          while(fromIndex=line.indexOf(findWord,fromIndex)>=0){
              fromIndex += findWord.length();
            output.collect(findWord, one);
          }
        }
       }
       public static class Reduce extends MapReduceBase implements Reducer<
Text, IntWritable, Text, IntWritable> {
         public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<
Text, IntWritable> output, Reporter reporter) throws IOException {
          int sum = 0;
          while (values.hasNext()) {
            sum += values.next().get();
          }
          output.collect(key, new IntWritable(sum));
        }
       }

       public static void main(String[] args) throws Exception {
         JobConf conf = new JobConf(WordCount.class);
         conf.setJobName("wordcount");
         conf.setOutputKeyClass(Text.class);
         conf.setOutputValueClass(IntWritable.class);
         conf.setMapperClass(Map.class);
         conf.setCombinerClass(Reduce.class);
         conf.setReducerClass(Reduce.class);

         conf.setInputFormat(TextInputFormat.class);
         conf.setOutputFormat(TextOutputFormat.class);
         FileInputFormat.setInputPaths(conf, new Path(args[0]));
         FileOutputFormat.setOutputPath(conf, new Path(args[1]));
```


SaaS 架构设计

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
        JobClient.runJob(conf);
    }
}
```

然后编译执行：

```
bin/hadoop jar /usr/alisoft/hadoop-installed/filesystem/wordcount
/wordcount.jar
org.myorg.WordCount /usr/alisoft/hadoop-installed/filesystem/wordcount/
input /usr/alisoft/hadoop-installed/filesystem/wordcount/output
```

执行完成后，查看输出结果：

```
bin/hadoop dfs -cat /usr/alisoft/hadoop-installed/filesystem/wordcount/
output/part-00000
奥运会        2304234
```

说明执行成功。

12.2 文件下载的分布式计算

一些大的互联网公司由于拥有很大的用户群，如即时通信服务、视频服务、在线游戏等。这些服务需要把文件资源下载到用户机器上才能使用，而且经常需要升级。由于传统的方式是由服务提供商采用服务器下载的方式提供文件资源下载，这使得服务提供商面临很大的网络带宽需求压力，以及要求部署高性能服务器来满足大量用户的下载服务，成本非常高。

为此，我们可以采用分布式下载和部署的方式解决集中式带来的问题。通过使用 P2P 的技术，可以充分利用用户的计算机资源。因为，若有用户已经下载了一个文件，那么后来者可以从用户那里下载文件，而不一定要从服务器下载。若用户机器很多，则后来的用户基本上不用从服务器下载文件，直接从别的用户机器上下载文件，也就极大地减少了服务器的带宽要求。

实用的 P2P 设计思想，主要分为两类：

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 一类是采用中心索引的方式；
- 另一类是采用分布式的结构化的方式，主要的算法有 CAN、Chord 等。

12.2.1 中心索引方式

该方式采用一个中心的服务器来记录文件索引信息。索引信息主要包括可下载文件以及该文件存储的 IP 地址信息。客户端要下载一个文件时，首先向中心服务器查询该文件所在的 IP 列表。中心服务器接收到查询后，根据客户端的 IP 地址返回存储这个文件的一系列经过网络路由选择过的 IP 列表。客户端得到 IP 列表后，进行尝试连接下载询问，由此得到可以提供文件下载的其他客户端列表。然后使用断点下载的模式，向这些客户端下载文件碎片。最后，将这些下载的碎片组装成一个完整的文件。主要的架构图如图 12-2 所示。

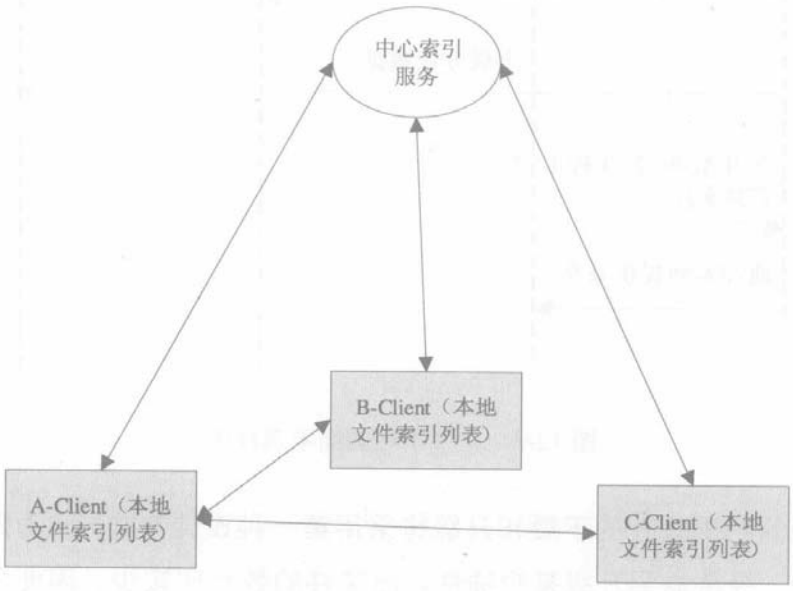


图 12-2 中心索引方式

Client 启动并进入稳定期后，向中心索引服务器注册，并通知中心服务器本地有哪些文件可以共享。中心索引服务器收到请求后，以文件为主键，文件的 Client IP 为值，在内存中建立索引列表。当 Client 退出时，通知中心索引服务器不再提供文件共享服务。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Client 文件下载的流程为：A-Client 需要下载某个文件 FileA，先得到该文件的种子信息（主要是文件大小及 MD5 校验码，以及中心索引服务器位置）；然后登录中心索引服务器，查找该文件的可供下载的其他 Client 的 IP 列表；接着与其他 Client 进行 P2P 通信，把文件分割成块，使用断点续传的方式从各个 Client 下载分块数据文件；最后合并成一个完整的文件。下载完成并校验通过后，通知中心索引服务器，本 Client 也可以提供该文件的共享下载的服务。文件下载的简单流程图如图 12-3 所示。

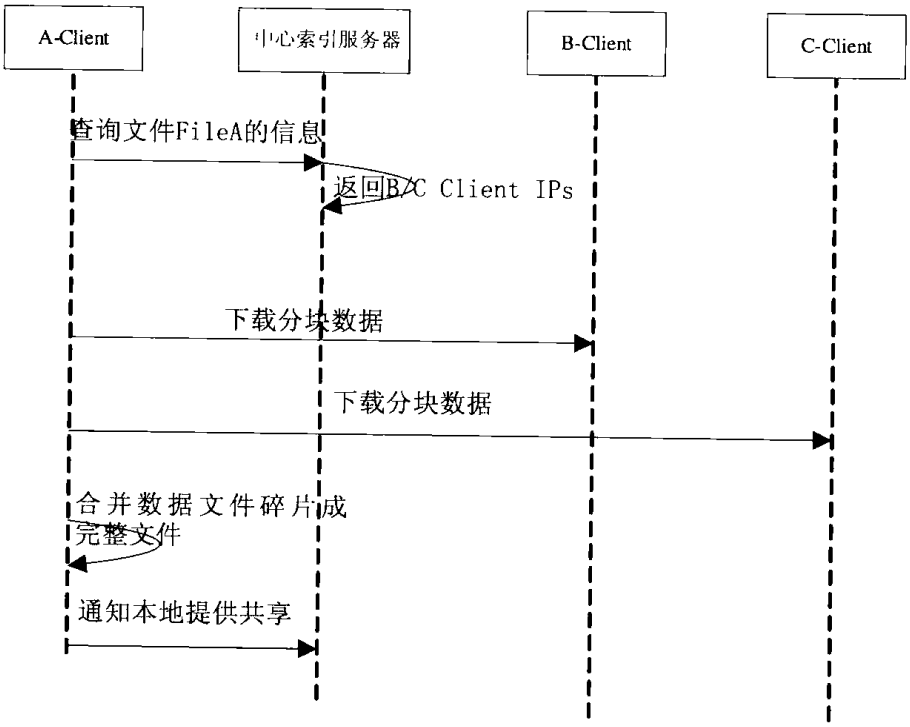


图 12-3　文件下载的简单流程图

阿里软件新的阿里旺旺的下载和升级将采用第一种设计方式。因为阿里旺旺的在线人数为几百万级，经常需要升级某个插件，而文件的数目比较少，因此采用中心索引的 P2P 方式非常合适。

12.2.2　分布式的结构化方式

该方式不存在中央索引服务器，但是它采用分布式散列表（Distributed Hash Table，

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

DHT）技术来组织网络中的节点。

DHT 是一个由广域范围内大量节点共同维护的巨大散列表。散列表被分割成不连续的块，每个节点被分配给一个属于自己的散列块，并成为这个散列块的管理者。DHT 的主要思想是：首先，每条文件索引被表示成一个（K，V）对。K 称为关键字，可以是文件名（或文件的其他描述信息）的哈希值，V 是实际存储文件的节点的 IP 地址（或节点的其他描述信息）。所有的文件索引条目，即所有的（K，V）对，组成一张大的文件索引哈希表。只要输入目标文件的 K 值，就可以从这张表中查出所有存储该文件的节点地址。然后，再将上面的大文件哈希表分割成很多局部小块，按照特定的规则把这些小块的局部哈希表分布到系统中的所有参与节点上，使得每个节点负责维护其中的一块。这样，节点查询文件时，只要把查询报文路由到相应的节点即可，该节点维护的哈希表分块中含有要查找的（K，V）对。这里面有个很重要的问题，就是节点要按照一定的规则来分割整体的哈希表，进而也就决定了节点要维护特定的邻居节点，以便路由能顺利进行。

DHT 模型下 Chord 实现算法借鉴的思想基本如下：

通过加密散列函数（通常为 sha-1），一个对象的名字或关键字被映射为 128 位或 160 位的散列值。由节点 IP（也可以是 IP 和端口）经过哈希得到它的描述，由资源关键字或内容经过哈希得到 K，其都为 m 比特（实际情况一般为 128），则节点描述和 K 的取值范围为 0～2^m。所有节点按照其描述符可以分布在一个逻辑圆环上，如图 12-4 所示。

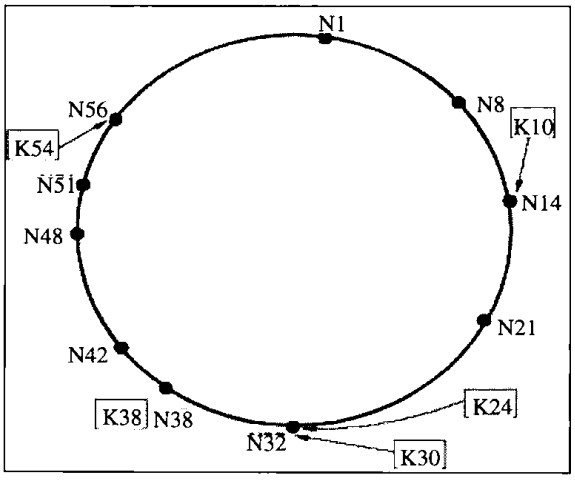


图 12-4　Chord 算法简图

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

资源索引的分布规则为：对于索引条目<*K*,*V*>，哪个节点描述的值最接近 *K*，则该索引条目就保存在哪个节点。同时每个节点要保存其最近两个节点的 IP 和端口。当客户端进行查找时，由 *K* 以及自身描述值决定是向上还是向下最近的节点发出查询请求，下一个节点收到后，如果自身没有该索引条目，则再次向下一个节点请求。如果网络内有 *N* 个节点，则这种查询算法的时间复杂度为 $O(n)$。于是 Chord 做了一步优化，其在每台节点上存储 *m* 个其他节点的信息，称为查询表。表中的节点不是相邻的，而是间距（节点描述）为 2^i（*i* 为 0～*m*）。这样，查询过程的时间复杂度变为了 $O(\log N)$。

对于以上纯理论模型，由于完全是对等点结构，运营商很难统计数据并进行控制。而且，即时优化路由的结构查找时间复杂度也为 $O(\log N)$，因此，这种结构最大的问题是难于维护。当一个节点退出时，不只是要通知整个系统清除其资源和把自身资源索引表转移给后面的节点，同时还要通知更新整个系统的路由表。作为一个健全的系统，肯定还要考虑节点非正常退出的情况，那么结构维护逻辑处理将更加复杂和难于实现。

12.2.3 改进模型

针对以上问题，现有一种改进模型，在系统中部署中心地址服务器，提供地址查询，该模型如图 12-5 所示。

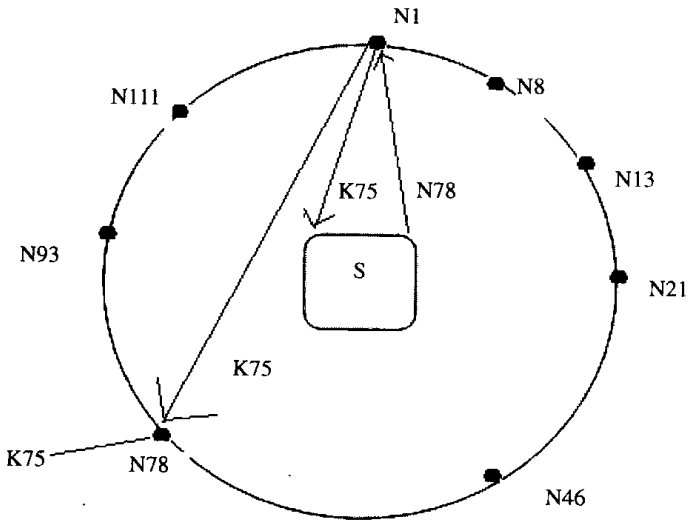


图 12-5 改进模型

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

在该模型中，中心地址服务器只记录节点 ID 和其 ID 端口的对应关系。由前面的介绍我们知道，K75 保存在比 75 大的节点集中最小的一个节点上，就是 N78。节点 N1 要获取 hash 值为 75 的资源，首先向中心地址服务器查询，中心地址服务器有 75 得知资源索引在 N78 上，返回给 N1 N78 节点的 IP 和端口。N1 拿到后连接到 N78，查询 K75 相关的资源提供机器。N78 返回之后，提交给业务层获取。

当新的节点加入时，有两个工作要做：

（1）从下家接管部分索引服务工作，并向中心地址服务器注册。

（2）发布自己的资源。

第一个工作，其先向中心地址服务器查询它的下家，然后连接到下家进行部分工作交接。第二个工作，把自己的资源生成索引，然后从中心地址服务器查询资源索引的保存者，把资源索引提交到相应的节点保存索引。

节点退出的处理是比较复杂的工作。节点退出分为正常退出和异常退出（程序崩溃、死机、断网等）。对于正常退出，其首先把自己提供的索引服务递交给它的下家，然后把自己提供的资源从资源索引服务提供节点中删除。异常退出的处理比较复杂，系统要提供额外保障措施提供系统的可用性。首先下家备份上家的索引资源，同时上家要定时和下家握手。当下家收不到上家的握手时，则自己增加上家的索引服务。具体处理是把备份索引转变状态，作为当前服务，同时向中心地址服务器注销上家。当然也有可能存在两个连续节点异常退出的情况，这种情况需要资源提供节点定时去检查。因为是属于全网成员的事务，所以这个动作不能够太频繁。可以设几十分钟定时检查，如果连接不到索引提供者，则把索引服务节点从中心地址服务器清除，并重新发布资源。

但是该模型还是有些复杂问题需要解决，如热点问题。对于升级文件这种应用，由于文件数比较少，大部分客户端都有文件的副本。如果通过这种方式使用 Hash 计算，则很容易产生热点，即所有的资源都是由少数客户端来提供下载，而这是不允许的。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

12.3 小结

本章简单介绍了两类分布式计算：MapReduce 和分布式文件下载。在实际中还有很多其他形式的分布式计算，这里不做详述。

对于有海量大小的日志文件，需要进行日志分析的系统，可以使用 MapReduce 分布式计算方式进行日志统计。

对于文件的下载和升级，在文件数目很多的情况下，使用第二种分布式文件下载方式合适，因为文件会被分散到不同的客户端中。

对于文件数目比较少（只有几万个），但客户端同时在线很多，达到百万甚至千万级的情况，如即时通信或在线游戏，那么采用中心索引的方式比较合适。

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第4篇

开放的 SaaS 平台

随着互联网基础设施的不断完善，业务数据的不断丰富，越来越多的互联网企业将长期积累下来的资源更深入、有效地利用，创造更多的价值。诸如 Google、Amazon、Facebook、Yahoo、阿里巴巴等互联网巨头，高举开放平台大旗，正在引领互联网时代的软件革命浪潮。

业界对于开放平台的最初定义，主要指互联网领域的网站从技术层面把自身数据通过 API 接口提供出来，供第三方开发商使用或在此基础上创建新的应用。现在，我们认为这只是一种狭义的开放平台。本书认为开放平台不只是开放 API，开放数据、开放接口、开放界面都应当是平台的开放服务。从广泛意义上来说，开放应该是能吸引更多的角色进入平台。他们一起在平台上运作，一起增值，使平台具备自我完善和发展的能力。

- 开放平台和 OpenAPI 的关系到底是怎么样的？
- 开放平台将如何形成自己的生态圈？
- 开放平台会给 SaaS 行业带来怎样的影响？

在下面的章节中，我们将通过对 OpenAPI 的价值分析和技术实现方式的介绍，与大家一起探讨开放平台下的 SaaS 发展之路，以及 SaaS 软件生态圈的建设等问题。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第13章 Open API

13.1 Open API 的商业价值分析 ····· 263
13.1.1 Open API 的出现、起源与现状 ····· 264
13.1.2 Google Maps API 带来开放潮流 ····· 265
13.1.3 Mashup 在 Open API 的时代 ····· 266
13.1.4 Open API 的分类 ····· 268
13.1.5 Open API 的系统架构与运营 ····· 276
13.2 Open API 相关产品 ····· 279
13.3 Open API 的实现技术 ····· 282
13.3.1 远程接口技术 ····· 282
13.3.2 远程接口设计最佳实践 ····· 289
13.4 Open API 与 API 托管平台 ····· 290
13.5 服务集成平台设计与实现 ····· 293
13.6 小结 ····· 297

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

最近，一封来自淘宝网店店长的信让郭靖有点烦。这位店长在淘宝网上经营着一个专卖电子产品的网店。另外，店长还有一个实体店。而且，店长也在用郭靖的软件管理自己的进销存。但是，最近网店的业务发展很快，网上的订单每天有几十单，于是，店长雇了一个小二来处理网店的业务。但在处理业务的过程中，小二却做得很辛苦，他需要把实体店库存中的信息手工录入淘宝网，还要把实体店的库存和网上的库存进行同步。并且，实体店和网店的客户折扣率、收货地址等信息也需要同步。由于小二的工作量很大，经常因为忙于两边的数据录入，而来不及回复顾客的询盘，因而丧失了许多商机。

店长在信中强烈希望郭靖可以解决这个问题。随后的一段时间里，郭靖收到了其他客户发来的信，需求非常相似。经过和杨康的讨论，郭靖确认了这个需求是应该尽快解决的问题。于是，郭靖马上展开调研，并很快开始了实质性改造工作。为了实现这些需求，他制定了几个方案：

A. 通过从网上自动抓取数据，实现两个系统之间的数据同步；

B. 与对方网站（淘宝网）谈合作，要求定时导出数据，实现同步；

C. 向对方网站（淘宝网）提需求，要求提供一系列开放接口，可以实时取回数据，实现同步。

以上方案的优缺点分析如表 13-1 所示。

表 13-1 方案优缺点分析

	数据质量	实施成本	用户体验	系统依赖性	数据实时性
A	低	高	低	高	低
B	中	低	中	高	中
C	高	中	高	高	高

综合考虑后，郭靖希望能采用 C 方案进行改造。

郭靖登录淘宝网，发现淘宝网在 2008 年 9 月 8 日正式上线的“淘园计划”中，就已经开放了很多淘宝网的 API。这些 API 可以实现淘宝网数据的获取和网店的业务操作，并且，这些接口很容易通过阿里软件的互联平台来调用。郭靖立刻认识到，这是一个全新的网络编程模式！于是，他马上投入了工作。

郭靖新设计的系统逻辑结构图如图 13-1 所示。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

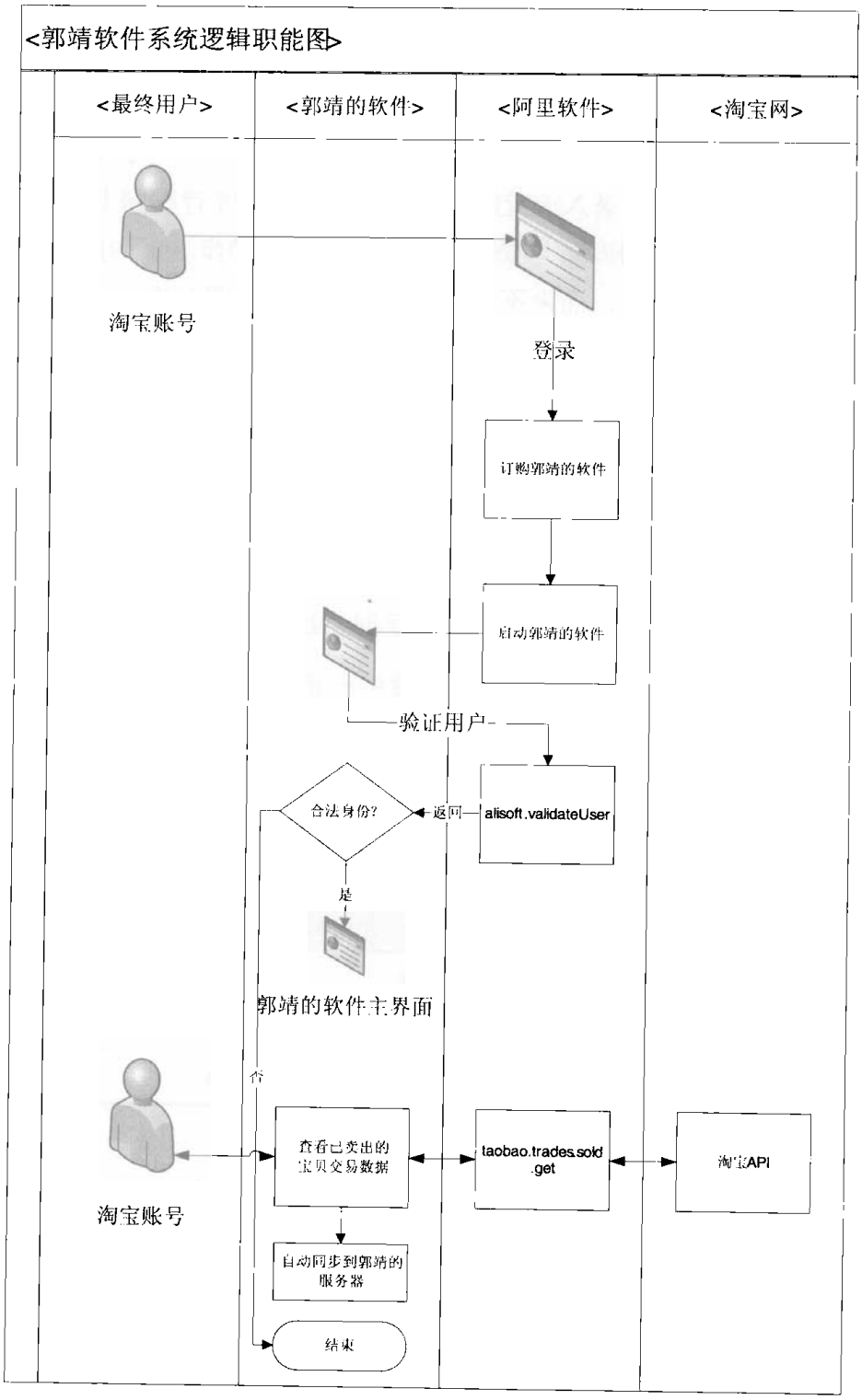


图 13-1　软件系统逻辑职能图

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

通过调用淘宝网的 API，郭靖的用户可以用淘宝账号直接使用软件。用户也可以使用原有的账号从原有的登录入口进入郭靖的软件（图 13-2 中 AEP 代表阿里巴巴软件互联平台）。

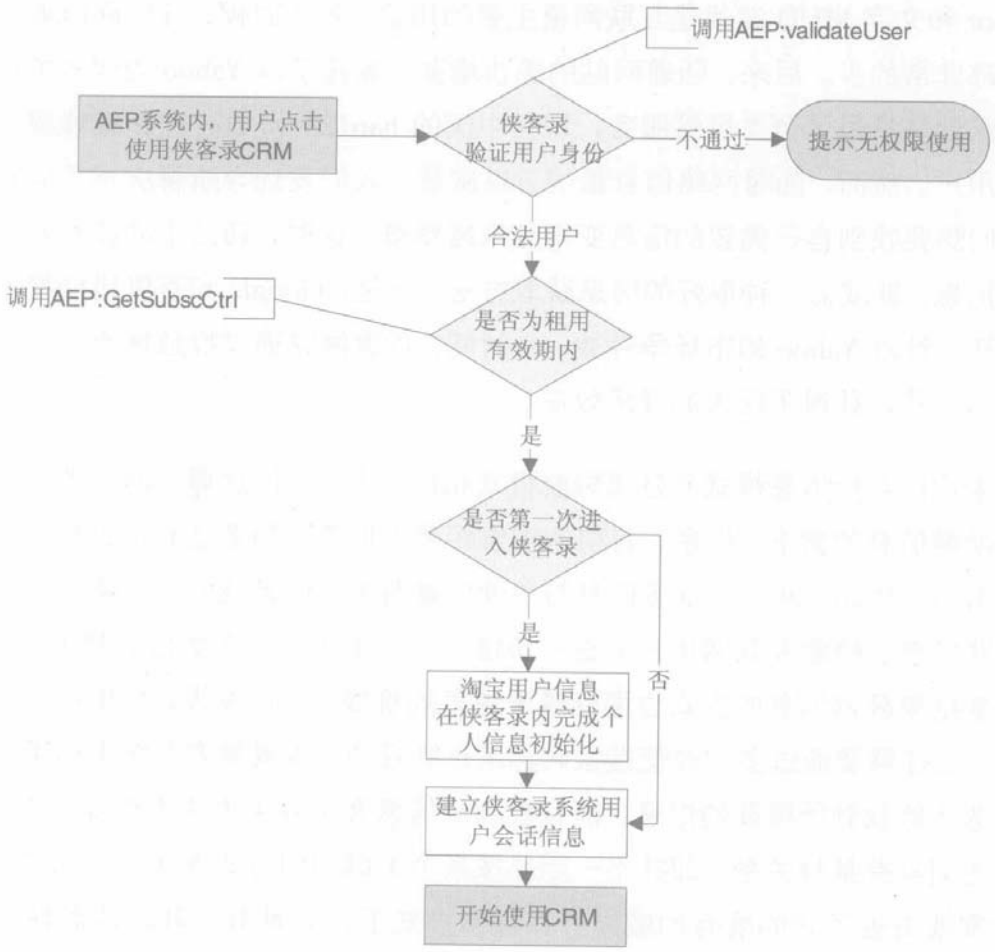


图 13-2　AEP 用户身份验证

13.1　Open API 的商业价值分析

Open API 一时兴起，各大巨头都希望通过 API 来吸引开发者，借此扩展自己的平台，以满足用户（更主要是自身平台用户）的多样化需求。在讨论 Open API 的设计实现前，我们先来了解一下 Open API 的发展历程以及由此带来的商业价值。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

13.1.1 Open API 的出现、起源与现状

在 1997 年前后刚刚开始上网的时候，人们常感觉到互联网上内容的缺乏。邮箱、Navigator 和文字 MUD 游戏是互联网最主要的用途。那个时候，互联网所能带来的信息和服务都非常的少。后来，随着网站的逐步增多，崛起了以 Yahoo 为代表的门户，其典型的特点就是提供了分类导航服务，现在中国的 hao123 和 265 还在继续服务着大量的互联网用户。然而，随着网络信息量的日益膨胀，人们发现导航解决不了信息太多的问题，人们想要找到自己需要的信息变得越来越艰难。这时，通过主动搜索网络来得到所需要的信息，就成为一种很好的信息获取方法。于是，Google 和百度借助搜索引擎的优势，展开了针对 Yahoo 的市场争夺战。一时间，搜索网站通过投放搜索广告和关键词竞价广告的方式，获得了巨大的经济效益。

搜索定位目标信息模式与目录导航模式相比，适应了信息爆炸时代人们快速精准定位找到所需信息的需求。但是，目前的搜索距离人们真正想要达到的目标，还是有很长一段距离的。比如，网上有很多信息与地理位置有关，但是我们却不能在地图上直观地看到这些信息。随着互联网用户的进一步增长，人类生活的各类信息都进入了互联网，通过搜索结果获取信息的方式也面临越来越多的挑战。人们发现，想要在搜索引擎上获取信息，往往需要通过多次改变搜索内容来达到目的，或者需要对搜索结果进行大量的人工筛选才能找到所需要的信息，而且找到的信息展示效果也并不直观。此外，很多信息彼此之间具有某种关系，却并不一定是按某个关键词的方式连接在一起的，在这个时候，搜索就失去了它的既有的威力。在搜索模式下，这种有关联的信息缺乏高效的关联检索。本来关联在一起的信息，就像一个个碎片，散落在各处，并不能很好地为我们所用。

2006 年开始，互联网兴起了 Web 2.0 的应用浪潮。这使得人们可以更方便地进行信息交互，人们在网上拥有的私有数据也越来越多。不同于传统的搜索和目录导航模式，Web 2.0 的网站更多的是提供了网络信息的整理服务。用户使用 Web 2.0 网站可以非常高效地找到某些信息，或管理一些信息，也就解决了很多信息连接过于分散的问题。

然而，随着 Web 2.0 网站的高速繁衍，人们在不同的网站上产生的大量信息（User

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Generate Contents）又呈现出新的分散趋势。人们又希望把这些 Web 2.0 网站的信息再进行组织，以满足传统的信息搜索需求和其他各种细分的需求。如何整理这些分散到不同网站上的数据，满足信息爆炸时代的信息获取的新需求，将是我们所面临的新挑战。要解决的问题主要有：

- 离散化的信息：不同网站间，信息与信息之间缺乏关联组织；
- 离散化的应用：我们所使用的各种服务存在于不同的网站上，我们需要来回切换网站，并且无法把两个应用的流程关联起来以完成一个任务；
- 离散化的身份：同一个人需要在不同网站注册账号；
- 离散化的关系：离散化的身份导致离散化的关系，在不同互联网应用上的人际关系或人与数据的关系往往是孤立的，新的账号建立，意味着要重建很多以前产生过的数据；
- 离散化的习惯：在一个网站上长期积累的用户习惯不能应用在另一个网站上，不同网站之间的用户习惯数据不能关联。

互联网由于信息量的暴涨，出现了目录模式的信息整理，随后出现了搜索模式的信息查寻，之后这些信息被进一步细化组织，形成了 Web 2.0 的模式。虽然，现在有这么多种信息的处理方式，但是面对信息爆炸时代，信息获取还是存在很多问题的。如何解决这些挑战，如何满足我们对信息获取的需求？Open API 的出现给我们提供了一种新的解决思路。

13.1.2 Google Maps API 带来开放潮流

在 Google 开放 API 之前，已经有 Amazon 和 eBay 等公司开放了 API。但是，它们的开放 API 集中在商品数据的获取上，与人们当时的日常广泛使用的互联网服务有一些距离。真正推行整个互联网开放理念的，应该是 Google 在 2005 年开放 Google Maps 的 API（见图 13-3）。由此涌现出众多的 Web 2.0 应用，从而也推动 Yahoo! Maps 开放 API。Google 的开放促使微软在 2005 年 9 月推出其“Web 平台”（Web Platform）策略，策略

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

中包括公开 MSN Search 等一些公共 Web 网站的 API，发布更好的开发工具等措施。微软的目标是采用 Web 2.0 或可编程 Web 的新兴模式，在已存在的、公开的 Web 网站基础上建立新的应用。

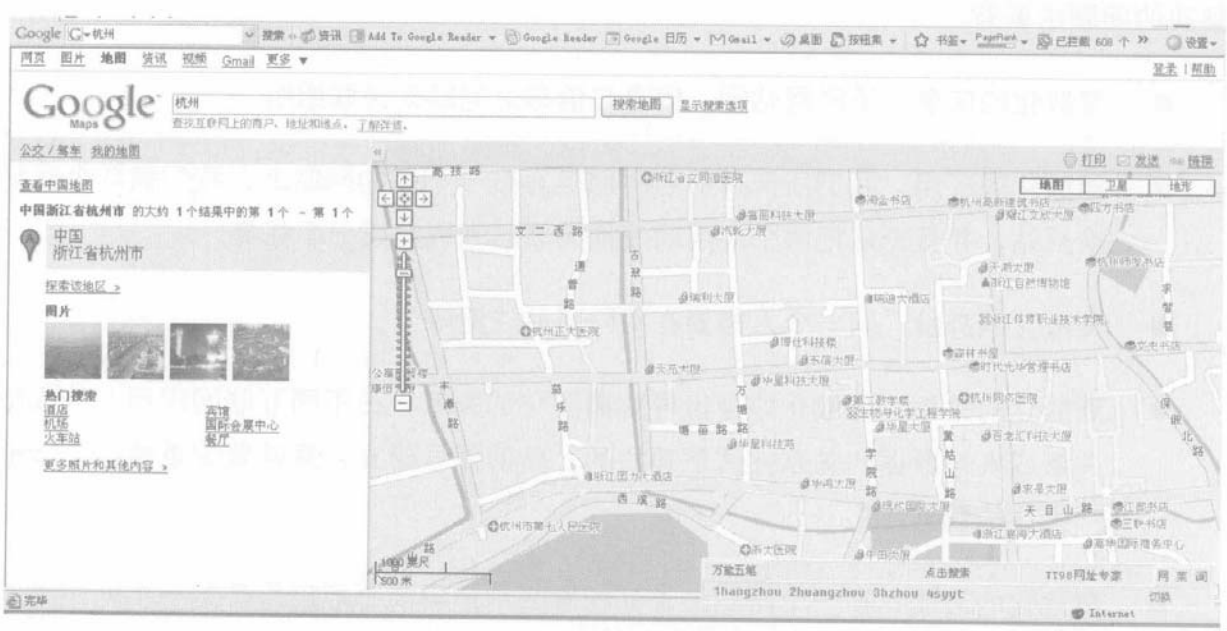


图 13-3　Google Maps API

在这种模式下，互联网公司不仅能提供对 Web 网页的简单访问，还可以将它们的 Web 网站转换为与操作系统等价的开发平台。2007 年，Facebook 推出开放平台，Facebook 在这个 SNS 平台开放了大量的用户数据接口，直接导致了互联网应用开放 API 的高潮。如今，Open API 正在被越来越多的用户所了解。

13.1.3　Mashup 在 Open API 的时代

在开放 API 的环境下，我们就可以对已有的一些离散化的数据进行重组，使其变得更有关联。这时候出现了一种叫做 Mashup 的概念。Mashup 又叫做 Web 应用“混搭”技术，指的是利用其他网站的 Open API 提供的内容进行重新搭配，从而制作出独特的、具有新价值的 Web 应用的一种技术。其中最具代表性的当属运用 Google Maps 提供的开放地理信息而创作出的令人眼花缭乱、极具创意的 Mashup 应用。比如，housingmap 是

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

一个非常 cool 的 Mashup 应用（见图 13-4）。它利用了 craigslist 和 Google Maps API，把它们巧妙地组合起来，让租房的分类信息有机地组织和直观地显示在 Google Maps 提供的地图之上。housingmap 提供了比单纯地图更多的租房信息，也提供了比单纯分类信息更好的信息展现体验。

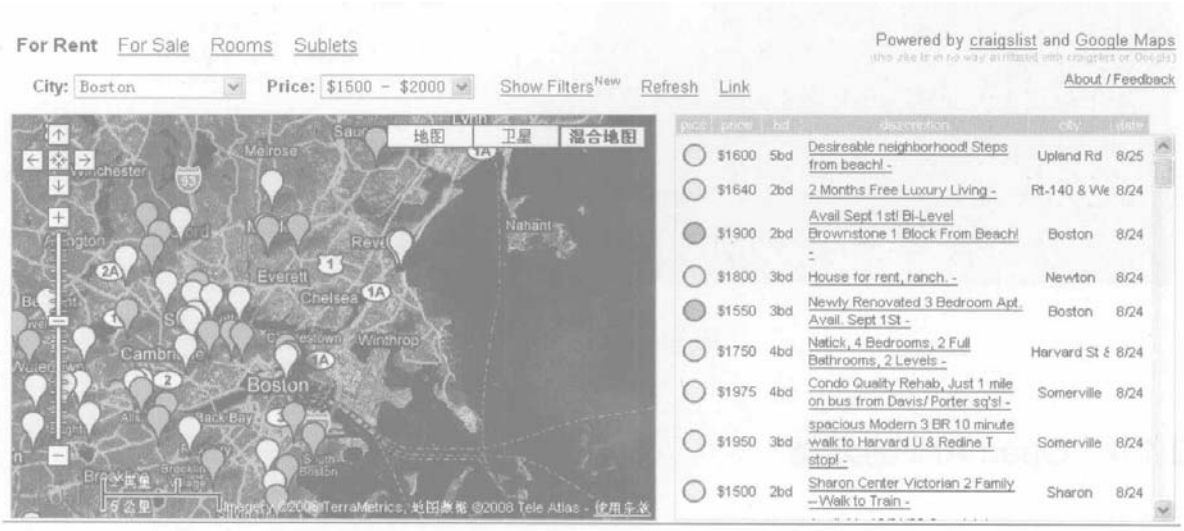


图 13-4　housingmap

twittervision 是把 Twitter 和 Google Maps 的 Open API 进行 Mashup 而形成的一个非常有趣的应用（见图 13-5）。进入 twittervision 网站，你会看到一个世界地图，地图会随着不断冒出的一些消息气泡移动，气泡里的内容就是当时某个用户在 Twitter 上发表信息的内容，而气泡的位置则是根据喊话人的地理位置信息而标记在了 Google Maps 上。目前 twittervision 是 Facebook 上的一个应用，提供了 3D 的地理展示。 在 twittervision 上漫游，可以了解世界各地现在正在发生的事。

Google 在开放 API 方面的确称得上是开拓者和领导者（http://code.Google.com/），不仅拥有 search API、chart API、map API、OpenSocial API 等一系列还在不断增长的 API 列表，更在开放平台方面发力，推出开放的手机平台 Android 和云计算平台 App Engine。在 Google、Yahoo、微软的带动下，Flickr、Youtube 等一系列网站也都纷纷对自己的服务提供了 API 供第三方开发者使用，社交网站 Facebook 等则为 Open API 提供了更多的 SNS 元素，让这些 Mashup 应用传播得更快。

SaaS 架构设计
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

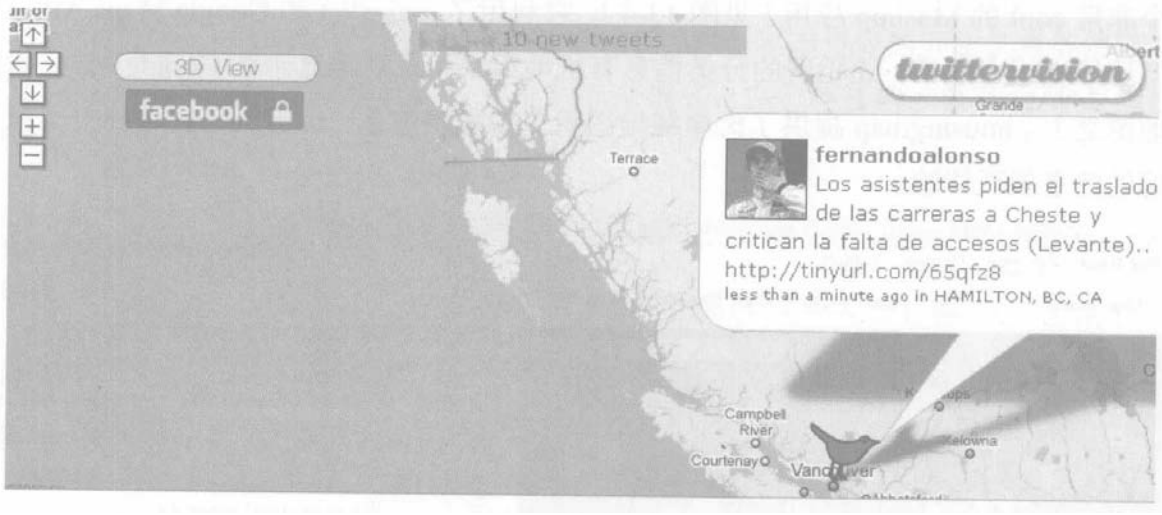


图 13-5　twittervision

13.1.4　Open API 的分类

Open API 正在大量的出现，新的编程模型逐渐形成。按照制定者（提供者）与遵守者（调用者）的关系这个维度看，Open API 可以简单划分成两个大类：

（1）专有类

一个 API 制定出来主要是为了制定者本身提供应用开发接口的目的，这样的 API 就叫做专有 API，如 Amazon、Facebook、淘宝、校内、51.com 等的 API。大部分的 API 都是专有 API。

（2）标准类

标准 API 主要是指具有以下含义的 API：制定者自己并不提供该 API 所定义的服务。例如，Google Open Social 是 SNS 的 API 标准；阿里软件的 SIP 平台是一种电子商务接口托管标准。此类 API 的发展目标是成为业界标准，或者该 API 已被业内很多服务提供者所遵守，例如 RSS 已经在互联网界形成事实标准。

Open API 按照提供的服务内容进行分类，大概可以划分成下面几类：

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（1）搜索类

搜索类的 API 主要由搜索引擎提供商提供，主要用来为网站添加搜索功能，或者利用其搜索结果进行组装。典型的例子如 Google search API、Yahoo search API 等。

运用搜索 API 进行 Mashup，可以做出许多很有意思的应用。比如，以下这个 URL：

http://www.langreiter.com/exec/yahoo-vs-Google.html

它将两个搜索引擎的搜索结果进行对比，也是非常好玩的应用（见图 13-6）。

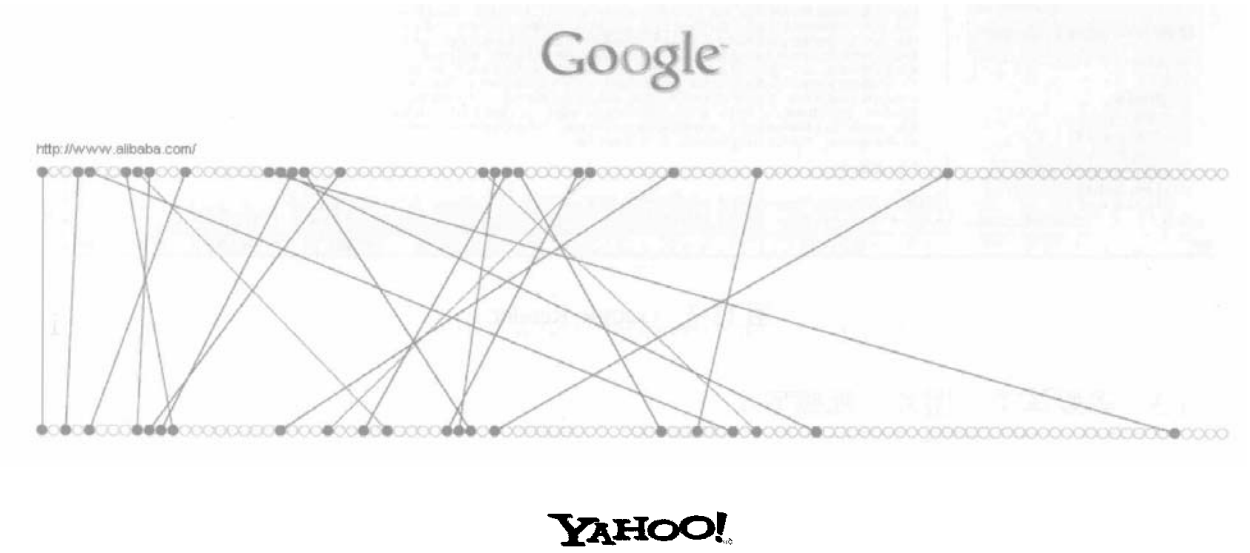


图 13-6　Mashup app

（2）文字资讯类

不可否认，RSS 接口可以成为几乎所有文字资讯类服务，如资讯类网站、博客、论坛的标准 API。因此，Google Reader（见图 13-7）、抓虾、鲜果等阅读器才能够让我们的阅读生活变得很轻松。

此外，还有许多专有的 API 提供这专有的文字资讯类服务。例如，Twitter 的 API、craigslist 的 API、豆瓣的 API 等。前 Google 员工创业项目 Friendfeed.com、加拿大华人的 planbus.com、国内创业网站 www.facefeed.cn 都是基于 RSS 或文字资讯类 API 的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

lifestream 服务。这些服务还能把普通 Web 页转变成简单的 API 提供者，通过自动分析页面的 HTML DOM 结构，提取出指定模板信息，或按 HTML 标准提取出统一的页面 Meta 信息。

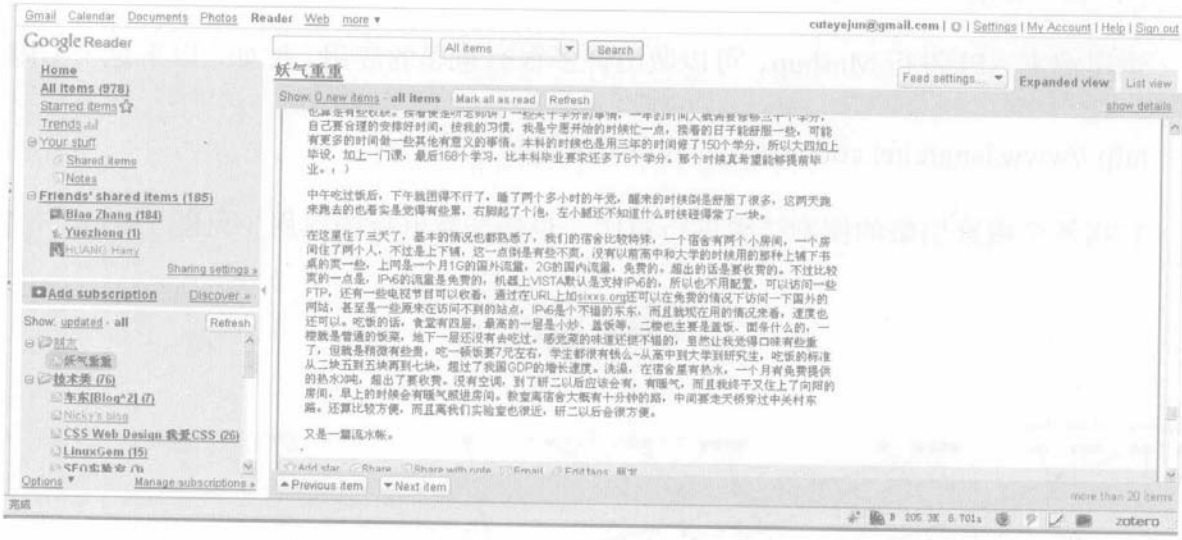


图 13-7　Google Reader

（3）多媒体类（图片、视频等）

Flickr 是雅虎旗下的著名图片分享网站，Flickr 开放了 API 接口，所以才有第三方网站开发者为它开发各种操作系统下的五花八门的工具。已经有大量的博客站点在引用来自 Flickr 和 Youtube 的图片、视频信息。

（4）地理信息类

在没有互联网以前，地图数据是很难得到的。互联网初期，地图数据也几乎没有引起人们的重视。随着汽车、旅游等产业的发展，人们对地图信息的需求越来越大。地图 API 开放之后，马上成为最有市场的 API。因为它为一维的文字信息提供了一个二维的载体，可以给人一种全新的互联网体验。此外，互联网上的信息大多能与地理位置产生联系，所以 Google、Yahoo 纷纷开放了自己的地图 API。Google 更是搞到了地球的卫星图片，做了三维的地理信息服务及开放接口，那就是著名的谷歌地球 Google Earth 及其 API。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（5）用户和关系类

随着网络社交的快速发展，SNS 用户量迅速增长。SNS 沉淀下来的用户关系信息和用户在 SNS 中的行为轨迹也逐渐成为了一类重要的数据。这方面的杰出代表就是 Facebook，它通过开放 API 使得大量的第三方开发者可以在其社会化网络平台上开发出成百上千的应用，从而极大地丰富了其平台自身的功用和乐趣。国内的校内网、51.com 也开放了一些用户关系类的 API。国内的海内和开心网也有大量的用户和关系信息的积累，但暂未开放这些用户和关系类接口。

2007 年 11 月 1 日，Google 宣布了自己的社会关系平台标准 OpenSocial。与 Facebook 专有接口做法不同的是，Google 致力于推广这套 API，力图使得全世界的社交网站都遵循这个标准 API 来开放它们的数据。凭借 Google 在互联网界的强大号召力，参与此开放平台的，除了 MySpace 外，还包括 Engage.com、Friendster、hi5、Hyves、imeem、LinkedIn、Ning、Oracle、Orkut、Plaxo、Salesforce.com、Six Apart、Tianji、Viadeo、XING 等网站。

在 Google 大力推广 OpenSocial 开放平台标准的压力下，2008 年 6 月，Facebook 把旗下的 F8 完全转变为开放源代码的平台，加入了开放 SNS 的大战之中。国内的多家 SNS 网站在开放 API 之前，一般都会参考 F8 的架构和实现。

（6）电子商务类

电子商务类网站提供的服务主要是围绕企业信息、商品信息和交易信息展开的。此类网站开放 API 的主要目的是，吸引第三方开发者开发各种附加功能，提高访问量、成交量或者用户粘性，从而提高网站自身的盈利水平；同时，将提高的部分盈利与第三方开发者利益分摊，吸引开发者持续开发和运营。比如，著名的亚马逊（Amazon）就是这方面的典型代表。此外，阿里巴巴集团旗下的 C2C 网站淘宝也已经开放 API。通过与阿里巴巴集团旗下另一家子公司阿里软件合作，淘宝网 Open API 平台得到了更多的技术和运营支持（见图 13-8）。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 13-8　淘园 TOP 平台

（7）注册类

还有一类重要的 API：注册类的 API。几乎所有的网站在开放接口的时候都会同时

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

提供一套供用户认证身份的专有 API，有基于一次性 Token 的，有基于 Cookie 的，有基于 SAML 的。

OpenID 项目致力于提供标准的、通用的注册 API，如果所有网站都遵守了 OpenID 规范，用户就不用为每个网站记住一套用户名和密码了。

OpenID 的基本过程如图 13-9 所示。

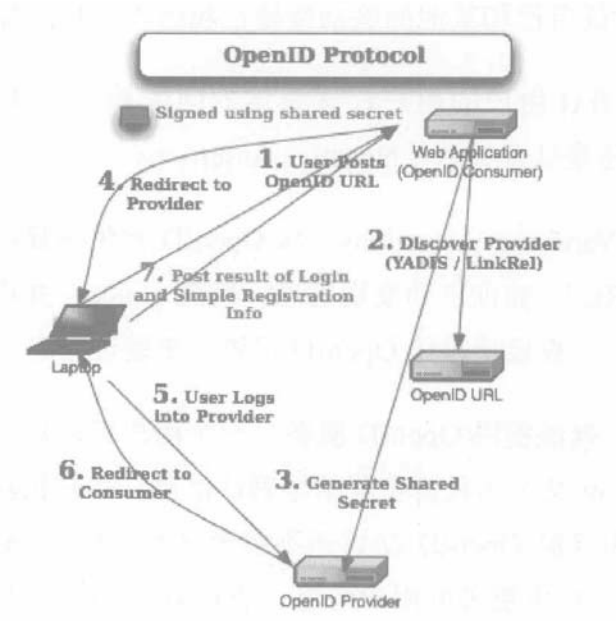


图 13-9　OpenID Protocol

① 用户通过他们的用户代理向依赖方提供一个 User-Supplied 标识，初始化认证过程（初始化）。

② 在规格化了 User-Supplied 标识后，依赖方执行自动发现来确定用户所使用的 OpenID 提供方终点 URL。需要注意的是，User-Supplied 标识也可以是一个 OpenID 提供方标识。

③ 依赖方（可选）和 OpenID 提供方建立一个 association（创建 Associations）——使用 Diffie-Hellman Key Exchange（Rescorla,E.,"Diffie-Hellman Key Agreement Method"）[RFC2631]密钥交换协议协商一个共享密钥。OpenID 提供方用 association 对消息签名，

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

同时依赖方验证这些消息；这就可以省略每次认证请求/响应的后续验证签名的直接请求。

④ 依赖方让用户的用户代理带着认证请求（请求认证）参数重定向到 OpenID 提供方。

⑤ OpenID 提供方确定用户是否有权并愿意接受 OpenID 认证。至于用户如何向他们的 OpenID 提供方验证自己和其他的鉴别策略已超出本文档所描述的范围。

⑥ OpenID 提供方让用户的用户代理重定向回依赖方，参数中指明是认证通过（Positive Assertions）还是认证失败（Negative Assertions）。

⑦ 依赖方校验（Verifying Assertions）从 OpenID 提供方接收到的信息，包括检查返回 URL（Return URL），验证自动发现信息，检查 nonce，并用在 association 阶段建立的共享密钥或发送一个直接请求给 OpenID 提供方来验证签名。

只要有一个域名，就能提供 OpenID 服务，一个用户可以拥有多个不同的 OpenID。而目标网站跟 OpenID 网站之间只需要简单地确认信息。用户仍然需要在目标网站上输入你的其他资料，因为目前 OpenID 协议还不包含这些资料。Google 的 OpenSocial 比 OpenID 的功能更丰富，提供更多的用户信息，使 Google 账号可以真正成为用户在网络上的身份证和通行证。在使用 OpenID 时，用户应尽量选择实力比较强的网站，可以确保 OpenID 不会因为网站服务不稳定，或者网站关闭而不能使用登录功能。

OpenID 从一定程度上节省了用户重复注册的劳动。并且 OpenID 的一个很明显的优势就是用户不必将密码泄露给目标网站，这样可以减少用户资料被窃取的可能性。大多数用户在不同的地方都用相同的密码，非常危险！因此，选择一个比较权威的网站作为 OpenID 服务是有价值的。

Yahoo 和 AOL 等网站已经支持 OpenID，但是众多的中小型网站却对这项技术没有投入足够的热情，因此 OpenID 远远没有达到统一用户的需求。

OpenSocial（见图 13-10）的出现在很大程度上解决了大家对 OpenID 的观望问题。因为 OpenSocial 一样可以解决用户的需求。什么是 OpenSocial？Google 认为：

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

"OpenSocial provides a common set of APIs for social applications across multiple websites. With standard JavaScript and HTML, developers can create apps that access a social network's friends and update feeds."

即：OpenSocial 提供了一套通用的 API，使得社会化应用可以在多个站点内复用。通过这个基于 JavaScript 和 HTML 的标准，开发者能够开发出很多小应用来访问社会化网络中的朋友信息和他们的状态更新信息。

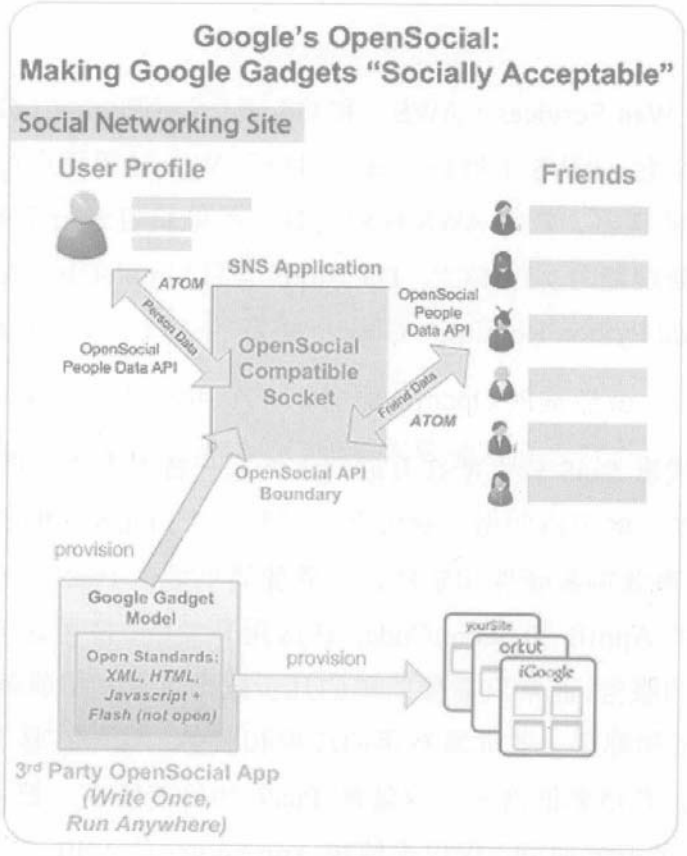


图 13-10　Google's OpenSocial

OpenSocial 由三部分 API 组成：用户的个人信息、好友关系和活动。对于好友的活动，从原理上比较容易实现。若不考虑隐私问题的话，只需要用一个 feed 就可以解决，各类中小型网站就可以方便地对这些 feed 进行编程了。

社会化网络是从最早的 BBS/Blog 兴起的，随着个人信息在 Web 上开始全方位的深

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

度呈现，除了文字之外，图片、音频、视频也大量出现在互联网上。在由个人产生的大量网络信息中，“关系”逐渐成为整理、导航和搜索这些信息的主要线索。因为，通过社会化网格，很多信息的获取比使用通用搜索引擎更快更精准。

如今，很多人在新的互联网环境下，重新审视了早期提出的社会化网络概念。这个概念已转换为更易普及的多个子集：SocialMedia、Social Computing（社会化计算）、社会化软件等。

（8）平台类

如今，Amazon Web Services（AWS）和 Google App Engine（GAE）是两个最为知名的开放 API 运行平台。AWS 上线后，马上引起了 Web 计算平台市场的重视。GAE 是一个运行 Web 应用的容器，它将 AWS 提供的许多类似特性进行了整体打包：存储类似 S3，自动伸缩性和处理能力类似 EC2，Datastore 类似 SimpleDB。App Engine 还提供了 AWS 没有的特性，如 Python Runtime、Google 特定的 API，以及可能是最吸引人的免费 Hosting 服务。

如果一个网站或者 SaaS 软件能在互联网上给客户提供开发（自定义）、测试、在线部署应用程序的功能，就可以叫做“提供平台服务”，即 PaaS（Platform as a Service，平台即服务）。ISV 不需要购买硬件和软件，只需要简单地在 PaaS 平台上创建一个应用，即可得到平台颁发的 AppID 和 SecretCode。在应用开发中，使用这些标识 ISV 身份的信息，调用 PaaS 提供的服务，通常只需要简单的几步鼠标点击和编码就能完成。利用 PaaS，ISV 能够创建、测试和部署一些非常有用的应用和服务，这与在基于数据中心的平台上进行软件开发相比，费用要低得多。这就是 PaaS 的价值所在。部署一个定制的数据库应用在 force.com 上的开发时间，仅仅是传统 App Server 的 1/40。平台类 Open API 体现了互联网低成本、高效率和规模化应用的特性。

13.1.5 Open API 的系统架构与运营

今天，Open API 作为互联网在线服务的发展基础，已经越来越多地成为互联网企业发展的必然选择。随着 Open API 的发布数量不断增加，它的存在也开始暴露出越来越

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

多的问题。

1. Twitter 的运营问题

2008 年 8 月前后，在网上看到这样一则有关 Twitter（见图 13-11）的消息：因为一台数据库服务器崩溃（原因是连接数过多）而导致很多关键功能被禁止。现在，技术问题已经成了 Twitter 进一步发展的最大瓶颈。



图 13-11　Twitter

2. 从 Twitter 得到的启示

从 Twiiter 遇到的故障中，我们可以发现一些问题，这些问题正是与 Open API 密切相关。Twitter 的整个应用模式建立在 Open API 的基础之上，它是一个提供 Web、IM、SMS 等消息服务的提供商。大量的文本消息通过即时通信软件（例如 Gtalk、MSN 等）、各种基于其 Open API 的客户端（例如 Twinkle 等）进行交互。在过去的一年中，Twitter 的服务中断次数和低质量服务问题正在随着 Twitter 用户数量的猛增而持续增加着。其中，已经出现了 Open API 的系统架构和系统运营之间的问题。

- Open API 是对系统架构的重要考验

从正确地理解业务和确定架构基础方面考虑，并对业务和系统架构做出合理的修正，才能使得 Open API 真正地实施好。大量远程访问的性能、数据的缓存、分布式的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

接口调用等，都是 Open API 系统需要不断优化的地方。

■ Open API 是对系统运营的重要考验

在运营推广之下，Open API 带来的并发访问量是常规的数十倍。必须在系统过程中不断优化代码，并且需要拒绝请求干扰系统的正常运行，还需要在系统运营过程中对系统架构进行反复地重构。

国内做 Open API 的难点在于极度缺乏优秀的系统架构和系统运营能力，使得 Open API 在性能上成为了产品的软肋。对于一个 Open API 的提供商而言，开放某一个或某一组 API 并非难事。但是，要持续地确保这组 API 一直可用，而且不会出现脏数据，这就是一件非常困难的事情了。同时，经常性的服务不可用，或大量垃圾信息和数据会严重伤害我们的用户。

从业务和运营层面看，我们可以得出 OpenAPI 新的划分维度：信息发布、消息通知、交易事务。

（1）信息发布

信息发布是指通过 Open API 将内容发布出去。从早先我们所说的 RSS 到现在新兴的 Amazon 的 S3 都是以这样的业务为基础的。通过这类 API，用户产生的数据和信息将会被多次使用，数据可以被第三方阅读和索取。这类 API 的特点是操作上以读多写少，内容变化不多为基础。通常一份内容会被利用很多次，而这很多次操作都是在读取它的内容。

（2）消息通知

通过 Open API 将内容分发出去。从早先我们所说的 SMTP、XMPP 到现如今刚提到的 Twitter 都是这样的业务。它的基础在于用户间的内容具有一定的定向性。它也会被多次使用，这样的使用包括内容的分发、阅读等各种操作。其特点在于整体上读写比例比信息发布要更平衡一些，更重要的是它的内容通常完全不会发生改变，只有增加、删除这样的操作。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（3）交易事务

就是说通过 Open API 完成一个事务型的交易。从早先提出的各种电子商务的 Web Services 到现在我们所知的 Amazon 的 EC2 都是以这样的业务环节为基础的。它的基础在于用户的操作是一个带有目标的一次性操作，这样的操作包括支付、系统调用操作等。它的特点在于所有的内容都是一次性的，需要即时知道操作的结果是否成功。更重要的是它的内容只在当时使用，之后均不可再用。

在准备提供一个 Open API 前，必须清晰地知道自己的 Open API 所支持的业务和运营环节是什么样的模型、什么样的用户量、多大的数据负荷和响应速度要求。对于这样的分析结果会在之后的系统架构中充分地体现出来。

系统架构是一个大家都能理解，但又往往做得不多的事。系统架构过程则是通过对业务环节的建模，以网络、硬件、系统、开发为基础，对要投入使用的产品进行一个综合设计的过程。系统架构更多的是要从成本、性能、可用性、可靠性等多个方面进行设计。通常来讲，对于以上三种业务环节的模型均有一系列的解决方案，这里不再多讲。

对于最后的系统运营，则是一个新鲜的事务。系统运营的工作，就是在系统为客户提供服务之后，以一个日常维护的形式保证系统的可用性、可靠性。其中，内容的可用性也是系统运营所需要保障的项目之一（在一些企业中也称之为产品运营）。Open API 作为一个服务，它所服务的用户是互联网上所有的用户。因此，必须在运行过程中对用户的所有操作进行记录、跟踪和分析，以保证系统及内容的可用性。在 Open API 设计和后期的不断重构中，也应保证接口调用方的系统稳定性。

13.2 Open API 相关产品

2008 年，国内 Open API 发展势头相当强劲。下面是几个已经涉足 Open API 业务的公司及其产品。

（1）搜狐博客的开放平台

SaaS 架构设计
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

以博客作为个人应用平台，由开发者及搜狐爱好者开发相关博客应用小程序；由搜狐推出相关文档说明。

网址：http://ow.blog.sohu.com

（2）谷歌 OpenSocial

目前，OpenSocial 由网络社区的广大成员联合开发。最终目标是，对于所有社交网站都能够执行API并托管第三方社交应用程序。许多网站在开发基于OpenSocial的应用，包括 Engage.com、Friendster、hi5、Hyves、imeem、LinkedIn、MySpace、Ning、Oracle、orkut、Plaxo、Salesforce.com、Six Apart、Tianji、Viadeo、XING 和天涯。

网址：http://code.Google.com/apis/opensocial

（3）淘宝开放平台

淘宝开放平台项目（Taobao Open Platform，TOP）是淘宝（中国）软件公司面向 ISV 提供 API 接口和相关开发环境的平台，包括 API 接口规范和定义、标准数据结构、API sandbox 在线开发测试环境。

网址：http://www.taobao.com/theme/tao_source

（4）天涯开放平台

由 Google OpenSocial 提供技术服务的天涯开放平台，主要用于商家整合，比如整合 7 天酒店。

网址：http://my.tianya.cn

（5）校内网开放平台

校内网已于 2008 年开始开放平台。邀请第三方公司及个人开发爱好者一起加入校内开放平台插件的开发，共同探索平台开放之路。

网址：http://www.xiaonei.com/developerIndex.jsp

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（6）聚友（MySpace）开放平台

MySpace 聚友开发者平台（MySpace Developer Platform，简称 MDP）是一个面向开发者的开放平台，开发者可以在上面开发并且发布符合 OpenSocial 规范的应用程序，比如 Widget、Gadget。

网址：http://developer.myspace.cn/community

（7）51.com 开放平台

51 开放平台 API 是采用 REST 基础的接口规范。所有的 51 开放平台 API 都是通过 HTTP 的 GET 或者 POST 请求来实现调用的。有了这些 API，你就可以通过你的应用程序获得 51.com 社交网络的用户资料、好友关系、照片等几乎所有可开放的数据。

网址：http://sandbox.developers.51.com/index.php

（8）康盛 uchome“我的空间”

UCenter Home 使用网站的开放平台应用开发标准（API），Manyou 将有机会帮助你与数万网站之间建立合作的纽带，为这些网站上的网民提供个性化的互联网应用。

网址：http://uchome.developer.manyou.com/uchome

（9）雅虎开发者社区

NCP 是中国雅虎针对站长天下等产品推出的开放式的 Web 应用/模块开发平台。开发者可以在该平台开发基于 Web 的模块，即 NCP 扩展应用。

网址：http://dev.cn.yahoo.com

（10）豆瓣 API

豆瓣 API 是豆瓣为第三方开发人员提供的编程接口。利用豆瓣 API，你可以在你的网站或程序中使用豆瓣的数据和功能。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

网址：http://www.douban.com/

（11）新浪魔方

新浪魔方通过组件化的功能组合，灵活方便地满足不同用户的个性需求。

网址：http://mofun.sina.com.cn

13.3 Open API 的实现技术

一项新技术的发展，需要更多的参与者，而不仅仅是服务的提供者。如果出现供大于求的局面，反而会把这些新的应用推向无价值的边缘。“开放”是目前整个互联网都关注的一个发展趋势。“开放”不仅意味着“分享”，更将意味着“共赢”。以上 Open API 基本上都需要解决的技术问题包括：调用者身份认证、数据访问权限、消息通信协议和业务数据格式这 4 个基础问题。本节主要讨论前两种技术。

调用者身份认证的确认，一般需要在 API 访问请求中加入调用者的签名，接口提供方验证调用者身份通过后，才能执行 API 的业务逻辑。

数据访问权限主要在业务层面进行控制。业务数据格式常用的有：SIMPLE XML、XML、JSON、TEXT、SOAP、REST。消息通信协议最常见的是 HTTP，也可使用 JMS 等消息机制进行 API 的对外开放。

13.3.1 远程接口技术

1. SOAP

SOAP 是针对 RPC 的一种解决方案，也叫做简单对象访问协议，很轻量。同时 SOAP 作为应用协议，可以基于多种传输协议（HTTP、SMTP 等）来传递消息。但是，随着 SOAP 作为 Web Service 的基础协议而被广泛应用之后，SOAP 被不断地增加许多新内容，使得现在开发人员越来越觉得 SOAP 很重，造成 SOAP 的使用门槛很高。在 SOAP 后续

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

的发展过程中，WS-*一系列协议的制定，增加了 SOAP 的成熟度，也给 SOAP 增加了学习成本和运行负担。

2. REST

REST 是将 HTTP 协议的设计初衷作了重新诠释。在 HTTP 协议被广泛利用的今天，越来越多的是将其作为传输协议，而非原先设计者所考虑的应用协议。HTTP 协议所抽象的 get、post、put、delete 就好比数据库中最基本的“增删改查”操作，而互联网上的各种资源就好比数据库中的记录。实际上，对于互联网的各种资源的操作最后总是能抽象成这 4 种基本操作。在定义了定位资源的规则以后，对于资源的操作通过标准的 HTTP 协议就可以实现，开发者也会受益于这种轻量级的协议。

使用 REST 进行接口设计需要注意以下几个问题。

（1）面向资源的接口设计

所有的接口设计都是针对资源来设计的。在这一点上，很像我们所说的面向对象和面向过程之间的设计区别，只不过将网络上的操作对象都作为资源来看待。同时，URI 的设计也是体现了对于资源的定位设计。后面会提到有一些网站的 API 设计说是 REST 设计，其实是 RPC-REST 的混合体，并非是 REST 的思想。

（2）抽象操作为基础的 CRUD

HTTP 中的 get、put、post、delete 分别对应了 read、update、create、delete 4 种操作。如果仅仅是作为对于资源的操作，抽象成为这 4 种已经足够了。但是，对于现在的一些复杂的业务服务接口设计，可能这样的抽象未必能够满足。其实，在后面几个网站的 API 设计中也暴露了这样的问题。但如果要完全按照 REST 的思想来设计，那么适用的环境将会有限制。因而，REST 思想也并非放之四海皆准的真理。

（3）HTTP 是应用协议而非传输协议

这点在后面各大网站的 API 分析中有很明显的体现。其实有些网站已经走到了 SOAP 的老路上，说是 REST 的理念设计，其实是作了一套私有的 SOAP 协议，因而可

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

以称之为 REST 风格的自定义 SOAP 协议。

（4）无状态，自包含

这点其实不仅仅是对 REST 来说的，作为接口设计都需要尽量做到这点。这是作为可扩展和高效性的最基本的保证。基于 SOAP 的 Web Service 也应尽量做到这一点。

关于 REST 和 SOAP 这两个技术在实现远程接口时的优缺点讨论已经有很多。本节从成熟度、效率和易用性、安全性、应用设计与改造这几方面进行分析。

（1）成熟度

SOAP 对于异构环境服务发布和调用，以及厂商的支持都已经达到了较为成熟的情况。不同平台、开发语言之间通过 SOAP 来交互的 Web Service 都能够较好地互通。不过，由于在部分复杂和特殊的参数以及返回对象解析上，协议并没有作很细致的规定，还是需要针对这些问题作部分修正。

国外很多大网站都发布了自己的 REST 风格开发 API，很多网站同时提供了 SOAP 和 REST 两种 Web Service。不过，根据部分网站的调查，REST 风格的使用情况要高于 SOAP。但是由于 REST 只是一种基于 HTTP 协议实现资源操作的思想，因此各个网站的 REST 实现都自有一套，在后面会讲述各个大网站的 REST API 的特点。也正是因为这种各自实现的情况，尽管在性能和可用性上会大大高于 SOAP 发布的 Web Service，但却在统一性和通用性方面远远不及 SOAP。由于这些大网站的 SP 往往专注于此网站的 API 开发，因此通用性要求不高。总的来说，SOAP 在成熟度上优于 REST，在企业级应用中使用 SOAP 的通用性更强。而在互联网应用中，由于 API 一般都比较简洁，且有大并发的要求，故使用 REST 更为合适。

（2）效率和易用性

SOAP 协议对于消息体和消息头都有定义，同时消息头的可扩展性为各种互联网的标准提供了扩展的基础。WS-*系列就是较为成功的规范。但是，也是由于 SOAP 为了满足各种需求而不断扩充其本身协议的内容，导致 SOAP 在处理方面的性能有所下降，同时在易用性方面以及学习成本上也有所增加。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

REST 之所以被人们重视，其实很大一方面也是因为其高效以及简洁易用的特性。这种高效一方面源于其面向资源接口设计，以及其操作抽象简化了开发者的不良设计，另一方面也最大限度地利用了 HTTP 最初的应用协议设计理念。而且，在笔者看来 REST 还有一个很吸引开发者的地方，就是能够很好地融合当前 Web 2.0 的很多前端技术来提高开发效率。例如，很多大型网站开放的 REST 风格的 API 都会有多种返回形式，除了传统的 XML 作为数据承载，还有 JSON、RSS、ATOM 等形式。这对很多网站前端开发人员来说，就能够很好地 Mashup 各种资源信息。

因此从效率和易用性上来说，REST 更胜一筹。

（3）安全性

安全性与成熟度密切相关。在当前的互联网应用和平台开发设计过程中，安全已经被提到了很高的高度。特别是作为外部接口给第三方调用，安全性可能会与业务逻辑一起受到特别重视。

SOAP 在安全方面是通过使用 XML-Security 和 XML-Signature 两个规范组成的 WS-Security 来实现安全控制的。当前，这些规范已经得到了各个厂商的支持，.NET、PHP、Java 都已经对其有了很好的支持（虽然在一些细节上还是有不兼容的问题，但是互通基本上是可以的）。

在安全性方面，REST 没有任何现成的规范可用。现在开放 REST 风格 API 的网站主要分成两种：一种是自定义了安全信息封装在消息中（其实这和 SOAP 没有什么区别）；另一种就是靠硬件 SSL 来保障，但是这只能够保证点到点的安全，如果需要多点传输的话，SSL 就无能为力了。安全这块其实也是一个很大的问题。今年在 BEA 峰会上看到有演示采用 SAML 2 实现的网站间 SSO，其实是直接采用了 XML-Security 和 XML-Signature，效率看起来也不是很高。未来 REST 规范化和通用化过程中的安全是否也会采用这两种规范，是未知的。但是加入 REST 的东西越多，REST 失去它高效性的优势也就越多。

（4）应用设计与改造

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

有时候，我们需要为一些已经存在的系统增加开放的接口，而开发人员的已有设计经验和思想会影响 REST 技术的推广。在资源型数据服务接口设计上，按 REST 的思想来设计相对来说要容易一些；而对于一些复杂的服务接口来说，非要按照 REST 的风格来设计可能会有些牵强。这一点其实可以看看各大网站的接口就可以知道，很多网站还要传入 function 的名称作为参数，明显已经违背了 REST 本身的设计思路，然而实现的效果也不错。而 SOAP 本身就是面向 RPC 来设计的，开发人员十分容易接受，所以不存在什么适应的过程。

REST 对于资源型服务接口来说很合适，同时特别适合对于效率要求很高，但是对于安全要求不高的场景。而 SOAP 的成熟性可以给那些需要提供多开发语言的，对于安全性要求较高的接口设计带来便利。所以，我们认为，纯粹说什么设计模式将会占据主导地位是没有什么意义的，关键还是看应用场景。

其实，最重要的一点就是不要扭曲了 REST。现在很多网站都跟风去开发 REST 风格的接口。不过，有的只是在学其形，不知其心，最后弄得不伦不类，性能上不去，安全又保证不了，徒有一个看似像模像样的皮囊。

现在已经有很多互联网公司提供了 REST 风格的接口设计。笔者参看了几个大型网站的 REST 风格的 API 设计，这里做一下说明。

（1）Facebook

- 请求消息

Facebook 在 URI 设计上采取了类似于 REST 的风格。例如，对于 friends 的获取就定义为 friends.get，前面部分作为资源定义，后面是具体的操作。其他的 API 也是类似形式，即：资源+操作。因此，就算使用 HTTP 的 get 方法都可能作了 update 的操作。其实，这已经违背了 REST 的思想。但是，在那么复杂的业务接口设计下，要通过 RUCD 来抽象所有的接口基本是不实际的。在 URI 定义好以后，还有详细的参数定义，包括类型以及是否必选。

- 响应消息

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Facebook 在响应消息设计上有多种方式，包括 XML 和 JSON。XML 有 XSD 作为参考，有点类似于没有 Head 的 SOAP，只不过这里将原来可以定义在 WSDL 中的 XSD 抽取出来了。

（2）Flickr

- 请求消息

```
http://api.flickr.com/services/rest/?method=flickr.test.echo&name=value
```

这里就可以很明显地看出它所定制的 REST 请求其实和 RPC 没有什么太大的区别。

- 消息返回

```
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat="ok">
    [xml-payload-here]
</rsp>
```

根据返回可以看出这已经违背了 REST 的思想，只是把 HTTP 协议作为传输承载协议，并没有真正使用 HTTP 协议作为资源访问和操作协议。

总的来说，Flickr 只是形式上去模仿 REST，自己搞了一套私有协议。

（3）Ebay

- 请求消息

采用 XML 作为载体，类似于 SOAP，不过去除了 SOAP 消息的封装和包头，同时在请求中附加了认证和警告级别等信息。

- 消息返回

类似于 SOAP 消息，不过删除了 SOAP 的封装和包头，同时在返回结构中增加了消息处理结果以及版本等附加信息。这个很类似于当前 Axis 2 框架的做法，将 SOAP 精简，同时根据自身需求丰富了安全、事务等协议内容。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

（4）Yahoo Maps

■ 请求消息

```
http://local.yahooapis.com/MapsService/V1/geocode?appid=YahooDemo&str
eet=701+First+Ave&city=Sunnyvale&state=CA
```

采用 REST 推荐的方式：URI+Parameters。

■ 返回消息

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:yahoo:maps"
xsi:schemaLocation="urn:yahoo:maps
http://local.yahooapis.com/MapsService/V1/GeocodeResponse.xsd">
 <Result precision="address">
   <Latitude>37.416384</Latitude>
   <Longitude>-122.024853</Longitude>
   <Address>701 FIRST AVE</Address>
   <City>SUNNYVALE</City>
   <State>CA</State>
   <Zip>94089-1019</Zip>
   <Country>US</Country>
 </Result>
</ResultSet>
```

SOAP 的精简 XML 返回，其他信息，例如出错码等信息由 HTTP 协议头来承载。

（5）YouTube

■ 请求消息

```
http://www.youtube.com/api2_rest?method=youtube.users.get_profile&dev
_id=YOUR_DEV_ID&user=YOUTUBE_USER_NAME
```

可以看到对于资源操作的 URI 定义也是参数的一部分。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 返回消息

```
<?xml version="1.0" encoding="utf-8"?>
<ut_response status="ok">
    <user_profile>
        <first_name>YouTube</first_name>
        <last_name>User</last_name>
        <about_me>YouTube rocks!!</about_me>
        <age>30</age>
        <video_upload_count>7</video_upload_count>
    </user_profile>
</ut_response>
```

这是自定义的类 SOAP 消息。

13.3.2 远程接口设计最佳实践

（1）请求消息设计

- 基本符合 REST 标准方式：资源 URI 定义（资源.操作）+参数。这类设计如果滥用 get 去处理其他类型的操作，那么和第 2 点无异；
- REST 风格非 REST 思想：资源 URI 定义+参数（包含操作方法名）。其实就是 RPC 的 REST 跟风；
- 类似于 SOAP 消息，自定义协议，以 XML 作为载体。（可扩展，例如鉴权、访问控制等），不过那就好比自己定义了一套 SOAP 和 SOAP extends。大型的有实力的网站有的采取此种做法。

（2）响应消息设计

- REST 标准方式，将 Resource State 传输返回给客户端，HTTP 消息作为应用协议而非传输协议；
- 以 XML 作为消息承载体，HTTP 作为消息传输协议，处理状态自包含；

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- 自定义消息格式，类似于 SOAP，提供可扩展部分。

13.4 Open API 与 API 托管平台

随着 Open API 的增加，API 的调用者需要一个统一的 API 寻址中心、一个统一的 API 帮助、测试环境和调用的 DEMO。John Musser 在 ProgrammableWeb 上已经聚集了近千个这类 API。为了方便调用者，这些 API 提供商需要提供以下信息。

ProgrammableWeb 是一个轻量级的 API 托管中心，而 Alisoft.com 提供了更为复杂 API 的 API 托管中心：Service Integration Platform（SIP）。Alisoft 的 SIP 提供了 Alibaba 集团下多家子公司的 Open API 托管，也提供了其他第三方 API 服务提供商的 Open API 托管。App 开发者可以在这里查找所需要的 Open API，使用测试 Console 进行测试，查看 API 数据格式和相关帮助，交流 API 使用经验。

据 2008 年 8 月的 ProgrammableWeb 统计显示，这些 API 被调用的网站数比例如图 13-12 所示。

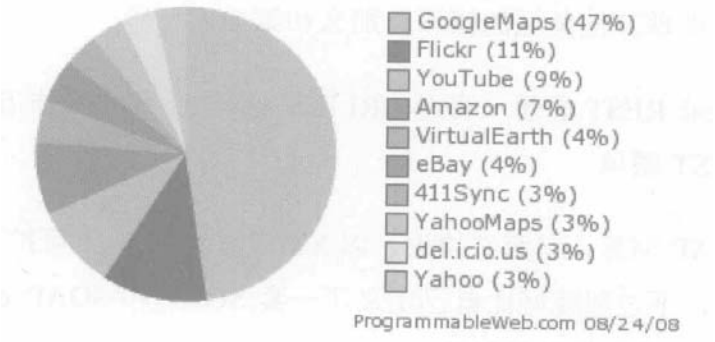


图 13-12　API 被调用的网站数比例

Google Maps API 在 ProgrammableWeb 被大量的 Mashup 应用引用，Flickr 的相册服务、YouTube 的视频服务也有大量的应用在引用。而从图 13-13 中看（http://cloudalicio.us/ tagcloud.php?url=www.programmableweb.com&x=19&y=6）这些 API 在时间维度上产生的影响并没有爆发式的增长。第一个原因是，引用这些 API 的

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

应用大多是为了好玩而不是商业目的创建的，有空闲时间的狂热者只会在晚上和周末创建它们，并没有打算用它们赢利。第二个原因是，像其他程序库一样，API 也是有一定学习曲线的。互联网公司当然希望通过最简单的方式暴露他们的服务，但并不是所有事情都可以简单地完成。所以 Open API 的价值最大化过程还需要时间，需要更多的互联网公司的投入，建立完善的服务机制，以方便应用开发者对 API 的调用，方便 API 服务提供者与调用者能得到双赢。

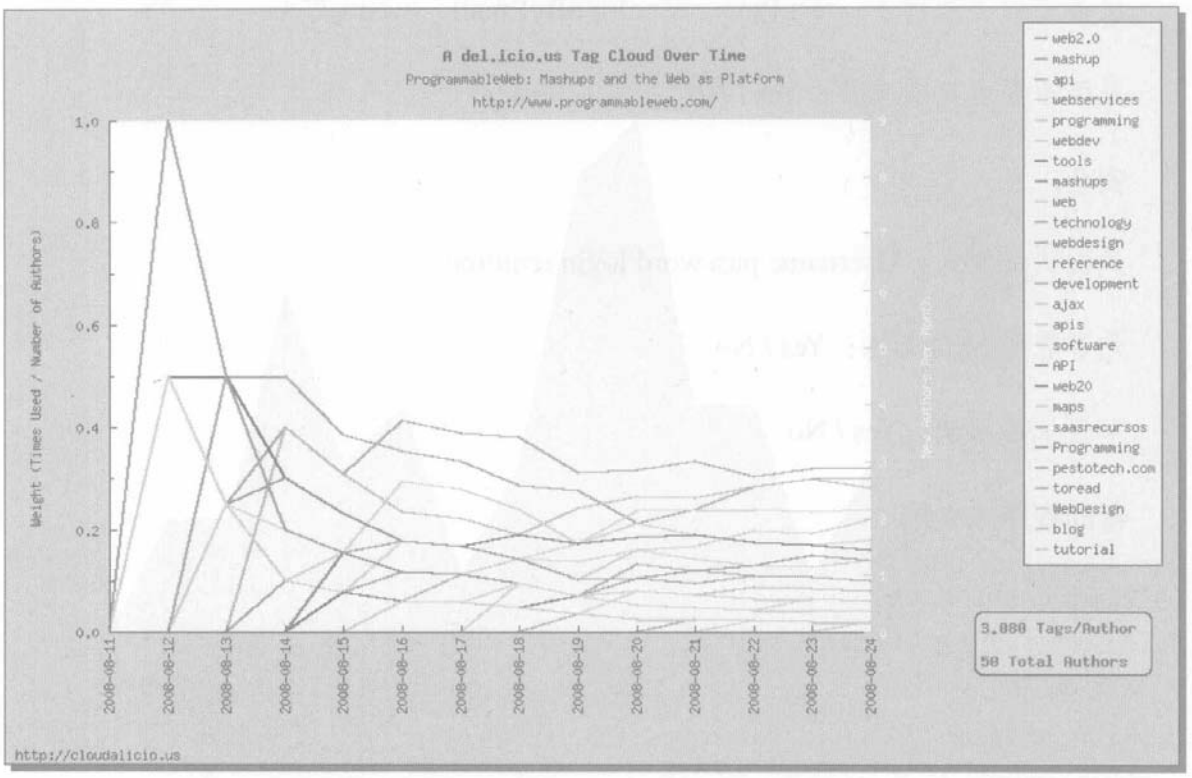


图 13-13　API 的调用增长趋势

ProgrammableWeb 上 Hosting 的这些 API，一般具有一些通用信息。在设计 Open API 时，可以从这些方面展开设计，调用者也可以快速地找到合适的 API，并快速展开调用的调试工作。

协议相关：

支持协议：REST

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

数据格式：JSON/TXT/SIMPLE XML/XML

服务器端 Service Endpoint：如 http://30boxes.com/api/api.php

功能介绍：

API Groups user、events、test

服务中的方法例子：test.Ping、user.FindByEmail、events.Get

是否需要客户端支持：Yes / No

安全：

身份认证模式：Username password login required

是否需要 SSL 支持：Yes / No

末登录前只读：Yes / No

帮助程序或文档：

```
Vendor API Kits
Community API Kits
API Blog
Site Blog
API Forum
Developer Support
```

注册和使用协议：

```
Signup Requirements API key required
Developer Key Required Yes
Account with Service Required Yes
Commercial Licensing Contact vendor
Non-Commercial Licensing
Data Licensing
Usage Fees
```


互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

```
Usage Limits
Terms of Service: General Terms of Service
```

在以下的章节中，将对 API 托管平台的典型代表：阿里软件服务集成平台（Alisoft Service Integration Platform）做详尽介绍。

13.5 服务集成平台设计与实现

下面是阿里软件服务集成平台接口调用的实例，通过这样一个请求，调用了淘宝网的 taobao.shopcats.list.get 接口。

```
http://sipdev.alisoft.com/sip/rest?sip_timestamp=2008-07-03+15%3A02%3A07&sip_appkey=10456&sip_sign=4AEA7243E585F83D31D3A1AA33D662CF&sip_apiname=taobao.shopcats.list.get&format=xml&sip_sessionid=a3fd257f802c41c1a009f1052084c778&v=1.0
```

在阿里软件服务集成平台下，API 服务提供商（ISP）与接口调用方（ISV）的部署关系图如图 13-14 所示。

服务集成平台的 API 接口调用安全问题，是通过签名与用户身份认证两种方式结合来解决的。对于大部分 API，由于其安全性需求很低，是一般的“读”类型的信息公开接口，这些 API 只需要有调用者身份签名即可。而对于淘宝网的部分“读写”接口，则需要用户身份的确认，以保证用户信息的安全。

对于需要用户身份确认的接口，所有进入第三方应用的软件都需要调用阿里软件官方提供的 ValidateUser 接口。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

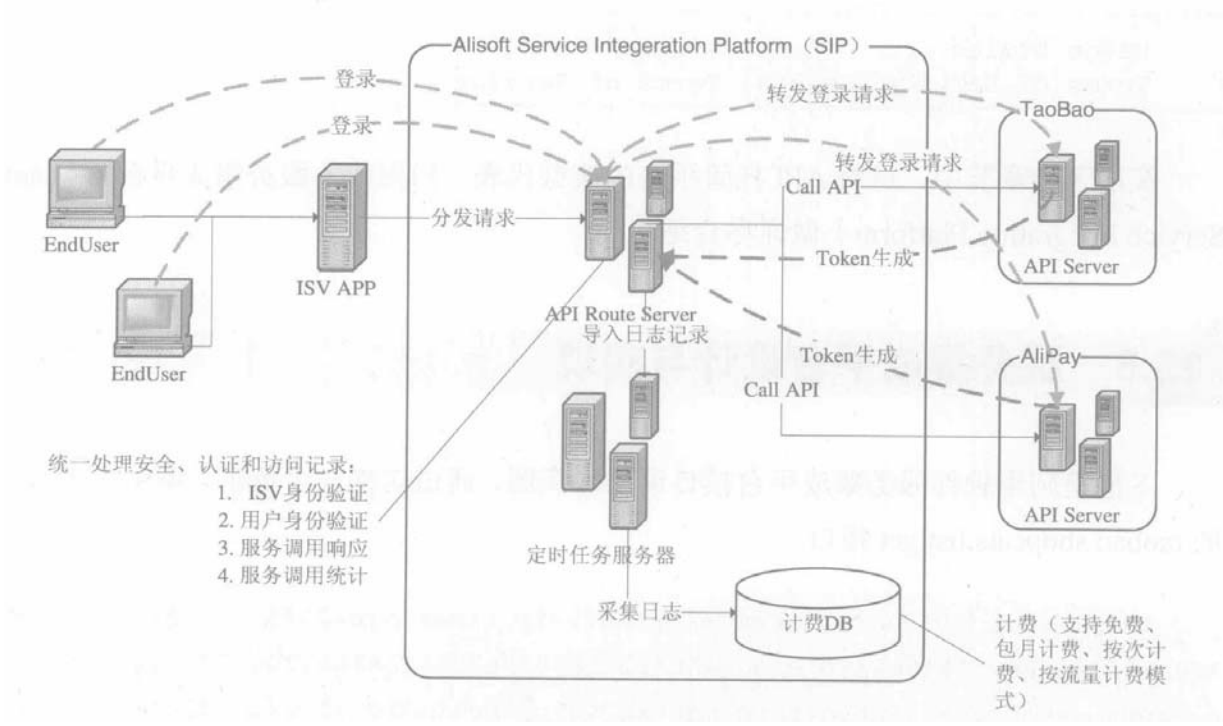


图 13-14　服务集成平台网络部署图

ValidateUser 接口的调用需要 SIP 接口调用文档中规定的系统级参数和应用级参数。系统级参数为 sip 开头参数，应用级参数就为免登录带过来的 3 个参数。调用需要的参数和每个返回值的说明，请查看接口列表，这里主要说明几个比较重要的参数。

- user_id：这是最终用户在 AEP 上的唯一用户身份 ID。拿到这个参数后，ISV 软件需把这个值记录到自己的数据库中，以便实现相关业务逻辑。比如，用户在 ISV 软件中产生的相关数据和这个值关联，实现展示这个用户数据的网页功能等。

- app_id：这是 ISV 软件的唯一不可变更的编号，注册软件时生成。可不记入数据库，但需保存在 ISV 自己的配置文件中，以后向 AEP 调用其他接口时需要用到这个值。

- app_instance_id：最终用户订购 ISV 软件生成的唯一编码。AEP 把同一次订购的人归入同一个 app_instance_id，因此 ISV 软件可以把同一个 app_instance_id

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

的人视为同一个公司或群体。使用这个参数是为了方便 ISV 软件形成人员组织结构。ISV 软件的用户表应该有一个字段保存这个参数。

- token：token 是身份验证令牌，可以在软件互联平台中跳转到 ISV App 时的 URL 参数中取得。注意：为了安全起见，平台提供的 token 只有 10 秒钟有效，并只有第一次提交有效，而且 token 每次生成值都不一样。

- sip_sign：平台是通过签名参数来防止传输过程中包被篡改的，所以这个参数在 SIP 接口调用中是非常重要的。SIP_SIGN 这个字段，为了保证调用接口的合法性和安全性，因此需要对提交的所有参数（sip_sign 参数除外）进行数字签名，并在调用的时候作为 sip_sign 参数传入。

在准备好接口调用的必备参数后，就可以开始调用阿里软件提供的免登录接口 ValidateUser 了。调用该接口其实很简单，就是发出一个 HTTP 1.1 POST 或 GET 请求到阿里软件认证中心。实例如下：

```
http://sipdev.alisoft.com/sip/rest?sip_appkey=10184&sip_apiname=alisoft.validateUser&sip_timestamp=2008-03-11
20:00:00&sip_sign=23ed99f1a0a1eac5f3787d522afde1c4&userId=111&appId=10184
&appInstanceId=
f2bf7b29-abee-4a3e-8678-949cf8bad9c7&token=0C1BC761C916D2A203D188993F2962
1D2D739299195A75B07CDC9A4E534C28F1B4886EDD2C47661DCAF33E5E2D4BD876
```

返回结果为 1，表示这是正常的软件订购人：

```
<?xml version="1.0" encoding="utf-8"?>
<String>
1
</String>
```

ISV 软件分析以上返回的 XML 数据格式，如果返回为 1，则完成一个软件正常的登录过程。这个 ValidateUser 接口视为用户是否已经登录过 AEP 的检查。ISV 软件在得到 1 这个值后，可能执行通常的 Cookie 建立、基本信息初始化等业务逻辑。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

签名参数生成：

sip_sign 生成的算法针对所有接口适用。简单地说，就是把所有需要的参数，先按字段 KEY 升序排列，再直接进行字符串连接组成这种串格式，即 CERT CORD 参数 1 值 1 参数 2 值 2…参数 7 值 7 参数 8 值 8 的形式，如果有一个参数 7 的值为空，则空值还是可以拼接到字符串上的，即 CERT CORD 参数 1 值 1 参数 2 值 2…参数 7 参数 8 值 8，再经过 MD5 加密后转成十六进制后得到一个 32 位的密文。sip_sign 参数 MD5 加密前如：

```
6c0a91f01a5411dd8179bad18f04aaceappId10256appInstanceId3a9b52ea-b1b9-
4431-9648-7f490f04a6dfsip_apinamealisoft.validateUsersip_appkey10256sip_t
imestamp2008-5-6 11:07:21tokenE3D04ABC7B8DD8B103D188993F29621D5C68ABE42FC
AC24E7CDC9A4E534C28F15240324753D62F01A00004DE2B18C4D7userId2536525
```

MD5 加密后如：

```
46E607E353FD66885BE452061A2B5190
```

字符串是由 CERT CODE 加上各参数名再加上各参数值组成的。注意：sip_sign 字段里用于签名的参数必须和接口调用 URL 后带的参数保持一致。除了 CERT CODE 和自身 sip_sign 外，其他都要保持一致。保持一致包括参数名和参数值。还有其中各个参数排列顺序是按照自然排序的，即按照参数的字母顺序排列。其中参数 sip_apiname、sip_sessionid、sip_timestamp 和 sip_appkey 说明如下：

- sip_apiname 的值是调用接口的接口名称，比如这里调用免登录接口 ValidateUser，sip_apiname=alisoft.validateUser。
- sip_sessionid，本次请求的 session_id。如果接口 URL 不传送此参数，那么签名里也不要对该参数进行签名加密。
- sip_timestamp，按照“yyyy-MM-dd HH:mm:ss”的格式来获取当前时间。
- sip_appkey，值为 appId。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

13.6 小结

本章介绍了 Open API 的商业价值和相关实现技术。

新的网络平台正在改变这个世界。基础模块就好比杠杆一般，可以在非常有限的时间内产生相当复杂、智能的应用。这些基础模块以开放 API 接口的形式提供给第三方时，形成了新的技术和商业模式。使用 Open API 构建基于浏览器及桌面的应用程序变得非常简单，开发者更加关注业务的易用性、功能健全性和自身业务的稳定性。

Open API 的实现技术主要是 SOAP 和 REST，这两项技术对于数据的开放提供了很好的基础设施。在这些基础之上，出现了一些对 API 提供更多辅助的框架平台，借助这些工具平台，Open API 的开放变得非常简单。最后本章通过对阿里软件服务集成平台的一个接口调用实例，向读者展示了服务托管模式下 Open API 的使用。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第14章 开放的SaaS平台

14.1 PaaS ……299
14.1.1 插件模式 ……301
14.1.2 七巧板模式 ……303
14.1.3 云计算模式 ……304
14.1.4 服务托管模式 ……310
14.2 互联提升价值 ……311
14.2.1 拓展软件的能力：聚合互联网资源 ……313
14.2.2 应用互联的价值链 ……313
14.2.3 基础设施提供 ……316
14.2.4 互联需开放平台 ……316
14.3 软件互联平台 ……317
14.3.1 软件大市场 ……318
14.3.2 软件生态圈 ……319
14.4 小结 ……323

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

侠客录 CRM 系统已经成功实施了一个自己开发的分布式数据库，并且这个分布式数据库随着 CRM 系统的不断改进，变得越来越稳定，性能也越来越高。尹志平听说郭靖有这个系统后，也想拿来“借用”一下，用到他的 SaaS 电子商务管理软件中，以节省一些技术部的工作量。

于是，杨康和郭靖对此事进行了讨论。

郭靖：把源代码给他们的话，我们这边程序更新了，他们会很麻烦。重新部署，重启服务器，新旧版本代码兼容性问题都会对他们的在线系统造成影响。

杨康：这种问题会让我们团队精力分散。我们能不能把在线的分布式数据库给他们使用？只要他们不能读取到我们的数据，不能修改我们的数据，不要拖我们服务器的速度。

郭靖：如果按你说的方法做，我们成本也很高。这种服务的提供需要投入大量的时间、人力和硬件去支持，需要有大公司背景才适合提供这类服务。如果我们有足够的资源，我确实想试试成为这样的服务提供商。这样我们就不只是一个 SaaS 软件提供商了，我们可以为大量的 SaaS 软件提供分布式数据库服务。

郭靖和杨康的这次讨论正是关于我们下面要讲述的开放 SaaS 平台。建设一个开放的 SaaS 平台的动机有很多，他们的这次讨论提到了开放一个分布式数据库给第三方 SaaS 软件使用。这种开放的服务是开放 SaaS 平台的能力之一。如果同时还提供在互联网上开发（自定义）、测试、在线部署应用程序的服务，以及存储、队列、计算等服务，那么这些公共服务就被称为平台即服务（Platform-as-a-Service，PaaS）。

14.1 PaaS

平台即服务（Platform as a Service，PaaS）是 SaaS 技术发展的趋势，PaaS 能给客户带来更高性能、更个性化的基础硬件和软件服务。平台即服务从产生至发展并成熟需要很长的时间，2008 年对于 PaaS 领域来说是里程碑的一年。软件的趋势是向基于互联网的自动化服务发展，并随之而来可定制、可扩展、灵活部署等要求，PaaS 其实就是这种

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

发展方向的更广阔的一部分。你如果想要忽略 PaaS，这几乎是不可能的。基于 SaaS 衍生的 PaaS 平台正在茁壮发展并成熟，传统软件企业面临 SaaS 和 PaaS 平台兴起带来的考验。

随着 PaaS 的发展，会诞生越来越多的基于互联网服务的基础设施，将大大降低技术门槛让应用更容易生成，间接带动更多的商业模式创新。尤其是资金花在软件和硬件的比例会降低，给初创公司带来更大的生存空间。种类丰富的平台服务架构在现有的 PaaS 上，也会促成生态链的形成；最后，如果有统一的规范约束服务调用，在同样环境下，就能解决技术整合的问题。

在 Windows 之前，DOS 的成功在很大程度上归功于 IBM，IBM PC 的附带销售，让微软可以不用付出很大的推广成本，就可以很自然地得到了用户，并取得收入，可以说那时的 IBM 是一个平台，但微软不是。

早期的 Windows 版本也还不能成为平台，更像是运行在 DOS 之上的窗口程序，对用户的吸引力完全没有苹果系统高，苹果系统更漂亮，操作更人性。但微软并没有输给苹果，原因就在于微软对软件付费理念的推广和第三方开发厂商的引入，甚至允许第三方厂商开发 Windows 的资源管理器，在当时这两种策略特别是付费理念开始时很令人不解，但这也就是微软最精明的地方，精华所在。

常见服务的平台化带来 3 种基本角色：

- 平台自身；
- 最终用户（End User，EU）；
- 独立服务提供商（Independent Service Vendor，ISV）。

只有两种角色的平台是不稳定且极其危险的，因为这样的平台没有独特的竞争能力，也没有服务的多样性，很容易被复制。现在每个行业都有同质化竞争，因为这些行业的服务模式单一，引进第三个角色或更多角色，可以大大提升平台的价值和竞争能力。更多的角色可以分担不同的职能，容易形成分工和产业链。

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Google 能称为平台，与 AdSense 和 AdWord 的推出有很大的关系。因为在没有这两个产品之前，Google 只有自己和最终用户两种角色。最终用户是受益了，但 Google 没有利益在里边，不符合平台的基本原则。只有这两个服务推出后，Google 平台才真正成为一个商业生态环境。现在的 Google，有 Google、AD 发布商、第三方 AD 投放的网站、最终用户 4 大角色。AdSense 的实现就是基于 Open API，一种非常浅的 API。API 调用者是第三方 AD 投放网，API 提供者是 Google。

在固有思维方式中，API 只不过是平台运作中一个重要的部分。Google 最初启动 AdWord 产品时，没有 Open API，照样运作得很好。所以，API 不是平台的唯一部分。在互联网行业，我们要建立平台，不是为了开放 API 而建立 API，而要以组建服务平台而开放 API。

因此，PaaS 是要将数据、API 和硬件都作为基础元件，真实的开放，吸引其他角色进入平台。这些角色要么作为利益输入方，要么作为最终用户价值实现方，一起在平台上运作，一起增值，使平台能在不需要自身推动的情况下，就具备自我完善和发展能力，形成生态系统。而要吸引其他角色进入，要作的工作也并不仅仅是提供数据和 API 接口这么简单。还需要在软件运行环境、软件扩展能力和软件运行的高性能方面提供大量的基础服务，这就是平台要解决的问题。

在 Open API 的商业化领域，存在 4 种不同的发展方向，它们在应用部署形态、开发模式、服务调用方式上有很大差别。下面我们来看一下这 4 种不同的方向。

14.1.1 插件模式

这种 PaaS 模式，可以进一步把支撑应用的下面几层功能，从中间件一直到数据库，还有虚拟化的 OS 环境，都通过网络出租出去。通过网络来进行远程开发、配置、部署，最后直接在提供 Hosting 服务的厂商的计算中心内运行。其中的典型代表是 Facebook F8 以及 Google 的 OpenSocial。图 14-1 展示了 HaaS 环境提供大量的 API 给第三方 SaaS App，这些 SaaS App 独立部署在自由环境中。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

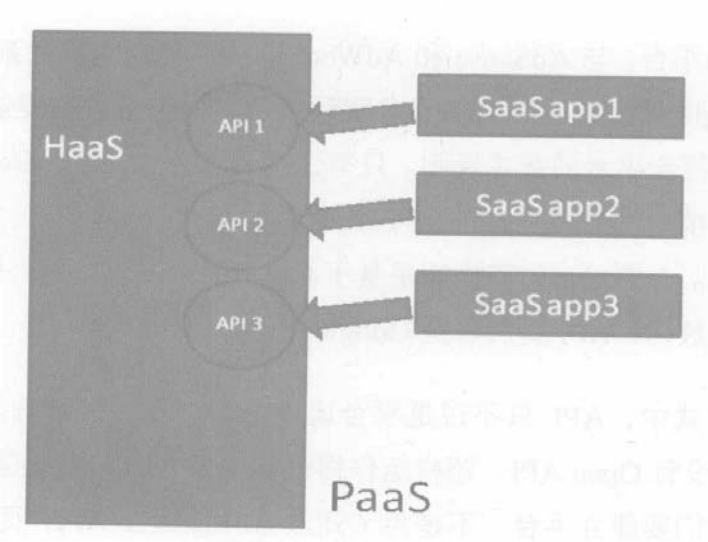


图 14-1　插件模式

在社交网站风靡的今天，用户关系信息也成为一大类重要的数据。这方面的杰出代表就是 Facebook，它通过开放 API 使得大量的第三方开发者可以在其社会化网络平台上开发出成百上千的应用，从而极大地丰富了其平台自身的功用和乐趣。F8，简单地说，就是以 Facebook 为平台，制作第三方软件。如果你觉得这听起来没什么，听听 Facebook 创始人 Mark Zuckerberg 对 Fortune 杂志说的一句话："We want to make Facebook into something of an operating system so you can run full applications, Zuckerberg told me, saying it would be analogous to the platform that Microsoft Windows provides for developers." 一句话：Facebook 将由此变成一个操作平台。

而 Google 也不甘寂寞，2007 年 11 月 1 日，Google 宣布了自己的社会关系平台标准 Open Social。并且与 Facebook 把接口专有化的做法不同，Google 致力于推广这套 API，力图使得全世界的社交网站都遵循这个标准 API 开放它们的数据。凭借 Google 在互联网界的强大号召力，参与此开放平台的，除了 MySpace 外，还包括 Engage.com、Friendster、hi5、Hyves、imeem、LinkedIn、Ning、Oracle、Orkut、Plaxo、Salesforce.com、Six Apart、Tianji、Viadeo、XING 等网站。

在 Google 大力推广 OpenSocial 开放平台标准的压力下，在 2008 年 6 月，Facebook 把旗下的 F8 完全转变为开放源代码的平台。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

两者的共同点在于 API 调用后的应用展现，统称为 Widget/Gadget（在我们看来没有区别）。OpenSocial 是在 Gadget 的基础上构建的，因此可以花费较少的服务成本（甚至无须成本）来创建一个出色的、虚拟的社交应用程序。使用 Google 小工具编辑器和一个简单的关键字/值 API，根本不需要服务器即可构建完整的社交应用程序。当然，如果愿意，也可以在自己的服务器上托管自己的应用程序。

Skype 是一个插件模式的开放平台。Skype 为第三方插件提供了集中式的软件选择界面，自由下载这些软件，并及时检查这些软件版本升级的功能，还提供了基础的 IM 通信和其他点对点通信的基础。Skype 和大量的第三方插件共同构成了一个开放的平台，这些插件一般是以 C/S 或纯客户端的模式开发的，因此它们不需要把程序 Hosting 在一个硬件平台上。这样的插件模式对硬件物理平台的要求比较弱，重在基础的软件服务功能上。

阿里软件的阿里旺旺 6.0 版本也是一种插件模式的开放平台。阿里旺旺和 Skype 一样提供大量的第三方软件集成，用户可以在这个插件平台完成很多操作，比如：选择软件，按月租为软件付费，自动完成软件升级等。旺旺 6.0 除了提供和 Skype 一样的软件层基础通信设施外，还提供了大量的服务器端支持。因为旺旺 6.0 不只是支持 C/S 和纯客户端的插件，还支持 B/S 配置型的插件，即一般的 Web 型应用也可以与旺旺插件平台通信。

14.1.2 七巧板模式

Salesforce 是“七巧板模式”的典型代表，这是一种以固定的数据类型和服务模块为主的开放平台。Salesforce 本身既是应用部署容器，也是应用开发车间，提供一些基础的模块供二次开发和包装。它成功因素之一在于 AppExchange 管理软件在线开发平台，基于这个平台，用户可以根据自己的管理需求，在运行期内完成定制化工作。有了 AppExchange，就可以更快速地扩展更多的应用。如今在 Force.com 上 Salesforce 创建了 5 万多个业务应用，如应收账款、员工招募、时间管理等。图 14-2 中的 SaaS App1 到 SaaS AppX 都存在于 Salesforce.com 的物理环境内，共享软件运行基础设施环境，最主要的特点在于模块化拼接。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

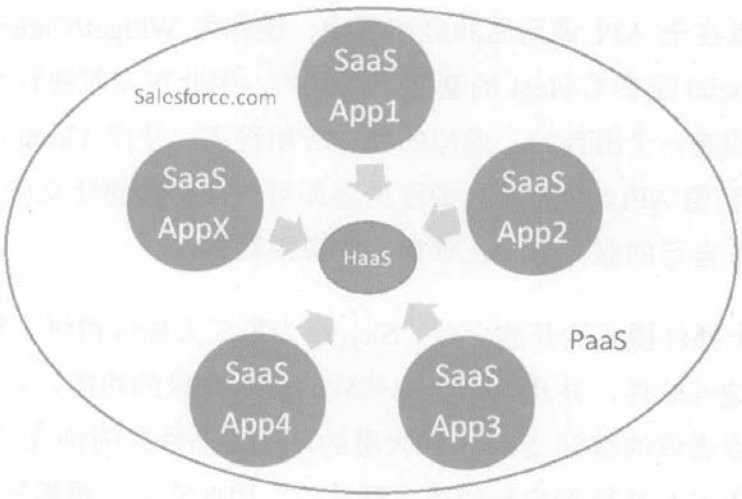


图 14-2　七巧板模式

据 Salesforce.com 官方报道，客户不只利用他们的 Force.com 平台集成 Google Map 这类 Web 2.0 网站的 Web Services 来完成 Mashups，更已经有客户利用它来成功集成了企业防火墙内的 SAP 应用。当租用 SaaS 的那些企业的 IT，对“服务”的概念和实践经验有了越来越深的领悟和掌握后，自然而然地将促进企业内更多非租用、自行维护的系统的服务化。

Salesforce 以自己的 CRM 产品为核心数据，专注于自身功能定制为主要发展方向。从今天的 PaaS 平台化的角度看，Salesforce 的模式显然是不够开放的，我们也认为它是一种初级的 PaaS 模式。“七巧板模式”还在不断发展，它是软件平台化历史过程的一条分支，已经比传统软件有了很大的进步，并对传统软件造成很大的冲击。

14.1.3　云计算模式

云计算是指用虚拟技术建造的数据中心或“超级计算机”，并以免费或按需租用方式把计算和存储资源提供给软件开发者，同时提供的还有软件运行的应用容器环境。我们可以认为这是一种硬件即服务 HaaS（Hardware as a Service）的模式。从一般意义上讲，主机 Hosting 就是 HaaS 模式的初级状态。但这里的 HaaS 还包括硬件之上的应用容器层。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

云是指互联网。和虚拟主机不同的是，云计算通过并行使用海量服务器，提供更强大的计算能力、存储和带宽。云计算面向的是软件开发者，而 SaaS 面向用户直接提供软件服务。云计算厂商通过硬件的随需应变虚拟化技术，让开发人员可以不用考虑硬件和带宽的制约来开发和运行应用程序，主要表现是分布式数据库、分布式存储、分布式计算等。图 14-3 展示了基于 HaaS 基础设施的云计算模式，在这一模式下，App1、App2 和 App3 可以集成服务并按需租用 HaaS 的部分资源，实现云计算应用与底层云计算环境自由集成。

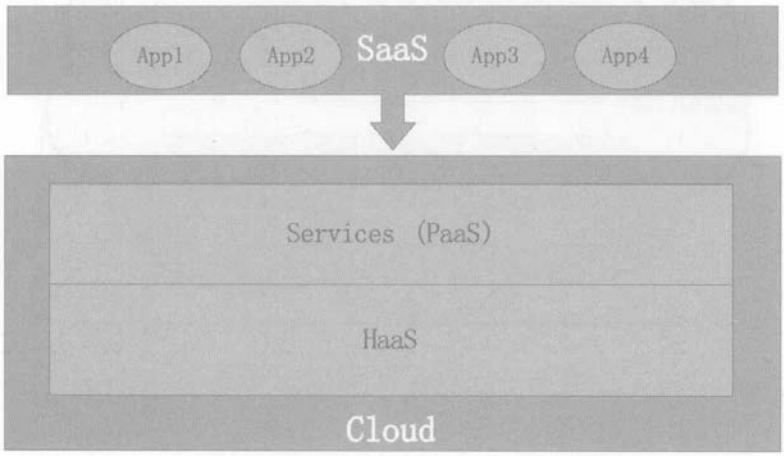


图 14-3　云计算模式

图 14-4 是从用户视角从外向内看 SaaS 和云计算的关系。

Amazon Web Services（AWS）应该算是第一个对 Web 2.0 / SOA 基础设施租赁市场的大规模尝试。今天，我们可以把 AWS 认为是云计算的平台。

Amazon 的 WebOS 策略是建立一个新的互联网计算模型。图 14-5 是我们对其理解的一个框架图。Web as a Platform 是 Amazon 的 WebOS 理念。图 14-5 的最底层部分也被称为云计算层，中间搜索服务层是基础的服务，最上层是应用服务。这些服务层都将提供大量的 Open API 给第三方开发者（ISV），这些 ISV 和 Amazon 在一起将形成一个统一的计算服务生态圈。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

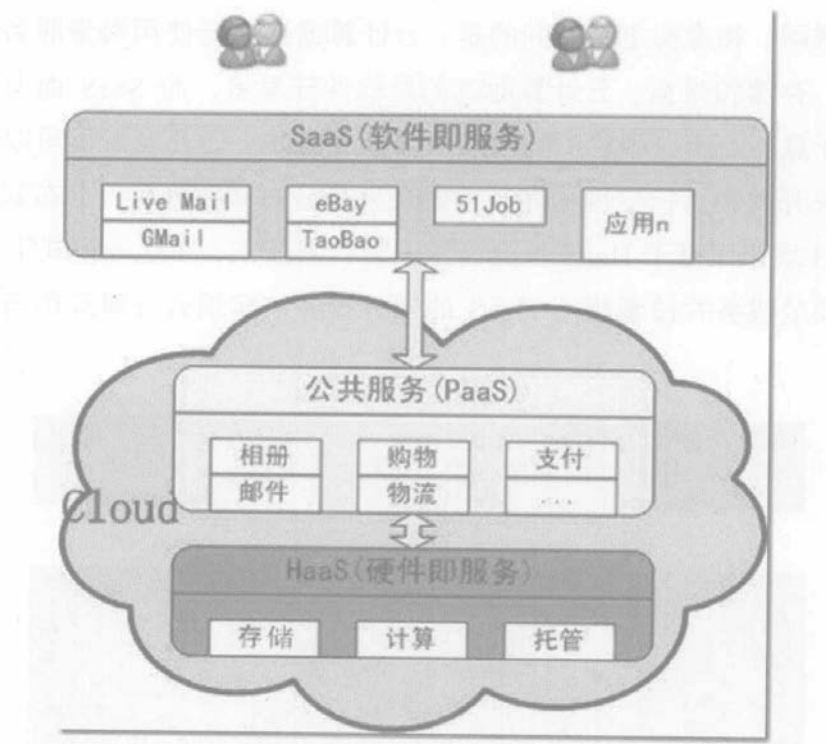


图 14-4　SaaS 与云计算的关系



图 14-5　Amazon Web Services Stack

通过 Amazon 的 WebOS 战略，这个西雅图的互联网巨人通过一套强大的 API 把自身的网站构架暴露于众。首先开放的服务是电子商务服务，该服务开放了亚马逊产品目

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

录主要产品的访问。该 API 相当强大，允许处理用户、收藏列表和购物车，当然最关键的是可以查看亚马逊的产品。

Amazon 以在线书店起家，成为全球领先的在线零售商。Amazon 在推出云计算之前收购了多家技术产品公司，之后推出了风格独特的云计算产品，也参与开创了云计算的商业模式。

Amazon 的云计算产品总称为 Amazon Web Services（亚马逊网络服务），主要由 4 部分组成，包括 S3（Simple Storage Service，简单的存储服务）、EC2（Elastic Compute Cloud，可伸缩计算云）、SQS（Simple Queuing Service，简单信息队列服务）和 SimpleDB。也就是说，Amazon 目前为开发者提供了存储、计算、中间件和数据库管理系统服务，开发者可以在 Amazon 云中开发应用软件，并基于 Amazon 的收费模式，向最终用户出租软件。

Amazon 最早推出的云服务是存储服务 S3，它提供无限量的文档、照片、视频和其他数据的存储，S3 已经被许多软件公司采用为客户提供下载、网络存储服务。

自从 Amazon 推出 S3、SQS 和 EC2 后，已经出现了许多有趣的增值应用。例如，有人用它来建构新一代的 B2B/EDI VAN，Doug Kaye 在这个 podcast 中，说明如何利用 AWS 的基础设施，设计出 GigaVox 这个 podcast 平台的架构。在 podcast 讨论中，Doug Kaye 除了讲解他如何善用 Amazon 出租的数据服务 S3、中介消息服务 SQS 和建立在 Xen 上的虚拟化平台 EC2，来设计出它的应用架构外，还谈到作为一个 early adopter，所遇到的种种限制问题如何克服，更提供了 Amazon 未来对 AWS 的改进参考。就像所有软件一样，这种直接通过网络租用的软件服务，目前仍属于非常早的阶段，当然还有不少可进一步完善的空间。这类的案例，非常有助于我们对此类服务的完备和成熟程度，进行更客观的理解。

顺带说一下，Amazon Web Services 起源于微软的 Internet Service Bus，从一个小规模、提供给开发者作实验性的环境和技术发展而来。

IBM 也有自己的“Blue Cloud”（蓝云）计划，但 IBM 还和谷歌联手将云计算提供给大学学生和研究人员。Google-IBM 云将会提供数百台计算机组成数个集群，主要来

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

自 Google 的定制机器和 IBM 的 BladeCenter 和 System X 服务器。谷歌和 IBM 两家公司也计划将它们的云合作拓展到企业级领域。

Google App Engine（GAE）是一种重要的云计算模式的平台。目前 App Engine 只支持 Python 语言，其他程序语言的支持还在讨论之中。PHP、Ruby 和 Java 是程序员最期望支持的语言。大量用户在这里讨论 App Engine 的相关问题和建议：http://code.Google.com/p/Googleappengine/issues/list。关于 PHP 的支持讨论也是异常的激烈：http://code.Google.com/p/Googleappengine/issues/detail?id=13。

GAE 提供了：

- 免费的主机空间；
- 免费的数据库 GFS；
- 免费的 Web 服务器；
- 完整的 Python 应用环境（目前版本为 Python 2.5）；
- 强大的技术支持（流量统计、免费域名、资源自由扩展）。

Google 还提供了一个软件开发包（SDK），用于让开发者进行本地化网站应用程序开发或上传已经完成的应用。

这个开发包包含：

- 一个 Web 服务程序，用来模拟 GAE 应用 Sandbox 环境；
- 一个本地版的数据存储方案；
- 本地模拟的 Google 账号集成；
- 支持使用 API 来分析 URL 和发送邮件；
- 这个开发包可以运行在所有安装了 Python 2.5 的机器上，并且支持 Windows、

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Mac OS X 和 Linux 系统。

因为这个开发包是以 Python 2.5 为基础的，所以你必须先要在你的机器上安装 Python 2.5（必须是 2.5 版本）。Mac OS X 10.5 用户可能已经安装了 Python 2.5，某些版本的 Linux 自带 Python 环境，但是版本可能比较低，需要升级成 2.5 版本。

GAE 是一个在线应用平台，是一个云计算的雏形，一个由 Python 应用服务器群、BigTable（Google 的一个研发的数据库系统）数据库访问及 GFS 数据存储服务组成的平台，它能为开发者提供一体化的、提供主机服务器及可自动升级的在线应用服务。开发者编写应用，Google 提供应用运行及维护所需要的一切平台资源。Google 推出的 App Engine 能让网络开发者更容易创建及升级在线应用，而不用在系统的管理及维护上花费精力。GAE 有以下优势：

- 使用 Google 的主机资源，运行第三方开发者的应用；Google 其实就是一个虚拟主机服务商，正在开拓一个全新的免费主机时代；
- 不需要传统的软硬件配置资源投入，GAE 提供一个完整的 Web 应用环境；
- 强大的可伸缩性，GAE 使你不用考虑访问用户的限制，从 1 个人到 100 万人，GAE 都能提供良好的支持；
- 免费开始使用，每个 GAE 应用程序可以使用 500MB 的存储空间，Google 没有具体限制流量和 CPU 的使用率，只是笼统地说明提供足够的流量和 CPU，可以支持每月 500 万左右的 PV。新启动的 Web 应用基本不用担心流量的问题了。

Windows Azure 是微软在 2008 年底推出的云服务操作系统（见图 14-6）。Azure Services Platform 提供第三方应用的 Hosting 环境，开放服务的管理。Azure 提供开发者按需订购计算资源和存储资源的能力，并可以按需扩展，自由管理开发者的互联网应用和云计算应用。Azure 支持一致性的开发体验，并可通过开发者熟悉的 Visual Studio 进行开发。早期开发的.NET 程序也将被这个平台支持。Azure 这个云计算的平台还将支持非微软开发语言的程序，并鼓励第三方工具和语言支持，如 Eclipse、Ruby、

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

PHP 和 Python。



图 14-6　Azure：微软的云计算平台

14.1.4　服务托管模式

以上 3 种模式，我们可以理解为独立的 ISP 提供服务，从服务访问标准到中间件，再到数据库和虚拟化的 OS/网络环境，都是各自独立的。现在有一种全新的模式，能够将不同 ISP 的服务，通过统一的标准和规范提供给 ISV，以降低服务集成时所带来的成本。这就是服务托管模式，如图 14-7 所示。服务托管模式下 SaaS 与 PaaS 的关系如图 14-8 所示。

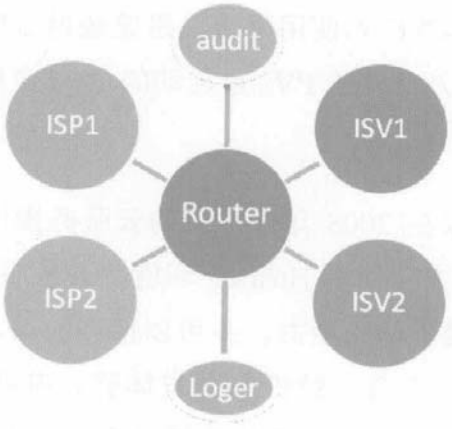


图 14-7　服务托管模式

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

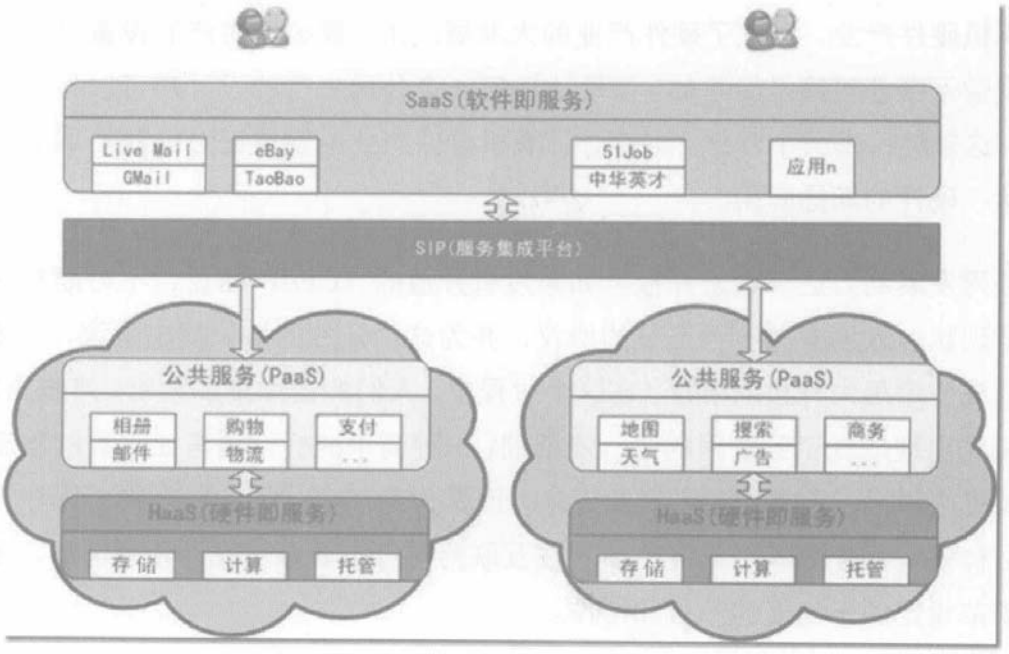


图 14-8　服务托管模式下 SaaS 与 PaaS 的关系

这个服务托管平台的典型代表，就是阿里软件 2008 年推出的 SIP（Service Integration Platform）平台。此平台的目标是构建一个面向开发者的 Open API 社区，ISV 和个人开发者能够在社区中，检索、学习、调用、测试符合自己应用场景的 ISP 提供的 API。

SIP 能够让 ISP 专注于数据服务设计，ISV 专注于应用/产品的开发，使服务能够在开放、协同、统一的环境中，与 ISV 应用更灵活、高效地融合，扩大整个生态圈的社会化价值，为构建基于云计算的网络架构打下基础。

SIP 为快速接入和交付服务、简化服务集成和管理，提供了通用的安全、计费、验证策略，并通过统一的 Sandbox 验证环境，为异构系统：Java、.NET、PHP 及 C/S、桌面应用提供了简约、标准化的接入解决方案。

14.2　互联提升价值

计算机硬件的发展有一个重要的动力：开放。开放的硬件接口标准让更多的厂商加

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

入了计算机硬件产业，促进了硬件产业的大发展。让一家公司生产的设备可以和另一家公司的设备无缝地对接是简单的，但是让任何一个公司生产的设计都可以与第三方的产品对接，这曾经只是一个梦想。如今，计算机硬件产业的发展已经向我们展示了一个美好的现状：硬件的即插即用。

互联网发展动力之一也是开放。如果没有开放的 TCP/IP 协议，不可能有整个互联网的通信现状。人们支持一种开放的协议，并为这个开放的协议提出建议，不断完善，最终会形成一个更为开放的协议。在这个过程中，人们的协作更加紧密，产物更加丰富。Web 网页的出现成就了互联网的第一波浪潮，一种简单的标记语言让无数的信息得到了传播。试想这种标记语言如果不开放出来，而是作为某个公司的私有物，那今天互联网会是什么样子？如今的人们已经习惯了在互联网网页中获取信息，分享信息，互联网的进一步繁荣肯定源于创新的开放和协作。

再看计算机软件，它曾经是一个产生信息孤岛的源头。大量的软件厂商开发出了丰富的软件，这些软件中存储、处理着海量的数据。然而，不管是同一公司的不同软件，还是不同公司的不同软件之间的通信问题都已经成为软件产业进一步发展的阻力。

将软件的功能、数据开放出来，形成服务，供第三方使用，就可以协作完成一些业务流程，从而产生更强、更丰富、更能满足用户需求的软件。让这些开放出来的服务以一定的标准加以规范，就可以让这些服务适用面更广，从而更容易被理解和使用。目前，软件业界一直在探索，为了让软件像硬件一样方便地“即插即用”，甚至向水电一样按需使用，业界已经提出了很多开放服务的标准，并且已经有一些实践应用。微软通过开放操作系统接口实现了个人电脑桌面软件的繁荣，互联网公司把软件和 Web 结合，把不同互联网服务器上的数据开放出来，协作完成一些跨网络的服务。但总体来看，软件的开放和协作还没有达到计算机硬件的标准。

基于互联网的软件实现广泛的开放和协作将带来一个更大的繁荣。我们相信开放与协作最终会使软件价值得到进一步提升。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

14.2.1 拓展软件的能力：聚合互联网资源

SaaS 模式使最终用户拥有了更自由选择软件的机会。当用户订购一个网上 CRM 软件后，一般马上就可以开始使用，不需要代码开发、硬件投入、网络维护。

软件通过聚合互联网的资源，调用其他互联网上的服务可以快速地完成自己的业务处理，增加软件的能力。我们可以设想某个 ISV 应用：

- 通过调用淘宝的搜索和类目接口展示宝贝；
- 通过支付宝的支付接口完成商品买卖；
- 通过 IP 电话或者视频接口，实现买家、卖家相互交流；
- 通过物流公司接口，实现商品配送；
- 通过 Google Map 或者短信接口，动态了解商品的配送过程。

……

SaaS 软件通过调用第三方的 Open API，可以实现快速的互联网资源集成、聚合，实现软件能力拓展。

14.2.2 应用互联的价值链

开放的 SaaS 平台应该构建一个面向开发者的 Open API 社区，ISV 和个人开发者能够在社区中，检索、学习、调用、测试符合自己应用场景的 ISP 提供的 API。通过 API 的调用，使得 ISP 和 ISV 之间的数据实现互联互通。在这个平台上，ISP 的资源和服务在开放的环境中深度集成、聚合，为 ISV 接入第三方服务，快速接入软件互联平台提供便利。各类角色在平台上找到自己的利益点，形成一条连续的价值链，从而共同推进平台的发展，产生更大的商业价值。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

业界一般认为应用互联的根本出发点在数据上，然而从商业模式角度看应用互联的出发点，就是吸引其他应用或角色进入互联的环境，大家一起得到利益，一起 Win-Win（双赢）。Google 开放的第一个目标是要吸引 AD 投放商进来，而且要买单。AD 投放商进来是因为这里有相当巨大的用户群体。为吸引更大范围的用户群，Google 提供了通用的搜索引擎，Facebook 则吸引第三方互联网应用开发厂商（ISV）进入平台，给用户开发出满足用户细分需求的网络应用。

ISV 为什么要进来?

因为它能在这里获取利益，不管是利益输入者提供的利益还是 ISV 通过最终用户获取的利益。

ISV 如何参与进来?

一种最高效的方式是通过 Open API。ISV 可以通过 Open API 获取的数据，从而与其他应用互联。

在这样一个价值链中，Open API 是应用互联的核心。与 Open API 相关的角色至少有以下五种：

- 应用互联平台（利益接收角色）：完成应用的推广和最终用户消费等功能；
- App 开发厂商（利益接收角色）：独立应用开发商；
- API 提供厂商（利益接收角色）：接口提供商；
- AD 投放商（利益输入角色）：广告投放客户；
- 最终用户（平台价值消费角色）：App 的租用者或广告的受众。

Open API 由 API 提供厂商（ISP）独立提供服务，或利用平台的基础设施提供服务。App 开发厂商可利用 Open API 实现业务，或单纯借助平台进行 App 推广。

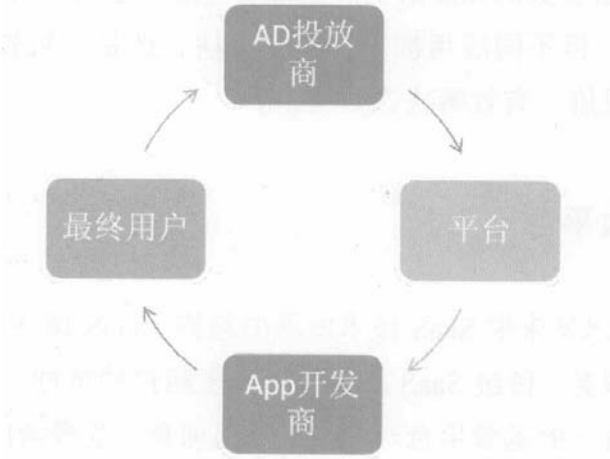
按 Open API 对调用者侵入程序，Open API 可以分成轻量级的 API 和重量级的 API

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

两大类。轻量级 API 对调用者程序侵入很少，调用者程序不严格依赖于 ISP，也不影响 ISP。重量级 API 对 ISP 有深入的关联，API 提供者的服务质量直接影响调用者的功能。类似于 Google AdSense 的广告嵌入到第三方网站属于轻量级 Open API。

互联网应用的互联环境中存在很多商业价值链，下面是典型的一条价值链：



在上面的生态链中：

- 对于平台：作为一个上下游产业链的中间者，主要是提供信息整理、信息推广和基础公共服务。平台以收取服务佣金作为利益。
- 对于 AD 投放商：广告投放也是一种投资行为，是为了利用平台的服务获取自己另一种业务的利益。
- 对于 App 开发厂商：允许在 App 页面上投放其他广告，这样就可以引入其他的广告投放商，让平台实现的一部分利益能分配到 App 开发厂商手中。App 开发厂商本身实现的业务功能可以服务最终用户。

在 Facebook 的应用互联价值链中，Facebook 建立广告投放平台和 F8 第三方应用平台；AD 投放商利用这个平台把广告投放到 FB 的页面，实现了平台的广告价值，也实现了平台利益的输入；最终用户点击广告，消费广告相关的商品，使 AD 投放商获得价值。应用互联的价值在 Facebook 的 F8 平台得到很好的体现。

SaaS 架构设计
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

14.2.3 基础设施提供

在应用互联过程中，各种应用都重复建设了一些模块：远程调用模块、分布式数据库模块、用户登录注册模块、用户收费模块、在线支付、用户访问日志统计模块和大量的第三方接口输入/输出参数的独立解析。虽然有些应用已经按 SOA 的思想实现了这些模块的松散耦合设计，但不同应用都实现这些模块，还是一种较大的重复建设。PaaS 正好提供了这些基础设施，有效解决这些问题。

14.2.4 互联需开放平台

开放的 SaaS 平台已是未来 SaaS 技术发展的趋势。SaaS 越开放，越能给客户带来更高性能、更个性化的服务。传统 SaaS 软件提供了多租户的特性，如果将不同的 SaaS 软件整合在一起，通过统一的运营平台对外提供产品服务，规模效应必将显现。这样的开放平台应该给客户提供在互联网上开发（自定义）、测试、在线部署应用程序能力，还能为 ISV 的产品开发和集成提供网络、存储、灾备、客服等一系列支持服务。图 14-9 描述了阿里巴巴软件互联平台的开放特性，上层的 SaaS 软件通过中间层整合复用了开放平台的公共基础设施，既减少了开发成本，同时也使 SaaS 软件功能更加丰富。

SaaS 软件由于天生的互联网特征，这些软件可以方便地与互联网其他软件通信，并且可以在线获取大量的互联网数据、服务等资源，从而实现软件能力的拓展。

2008 年对于开放的 SaaS 平台领域来说是里程碑的一年，因为开放概念在这一年里变得越来越清晰。软件的趋势是向基于互联网的自动化服务发展，并随之而来可定制、可扩展、灵活部署等要求，开放的 SaaS 其实就是这种发展方向的更广阔的一部分。你如果想要忽略这种开放和集成，这几乎是不可能的。基于 SaaS 衍生的运营和服务支撑平台正在茁壮发展并成熟起来，传统软件企业面临 SaaS 平台兴起带来的考验。开放的 SaaS 平台还将进一步提升软件的能力。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

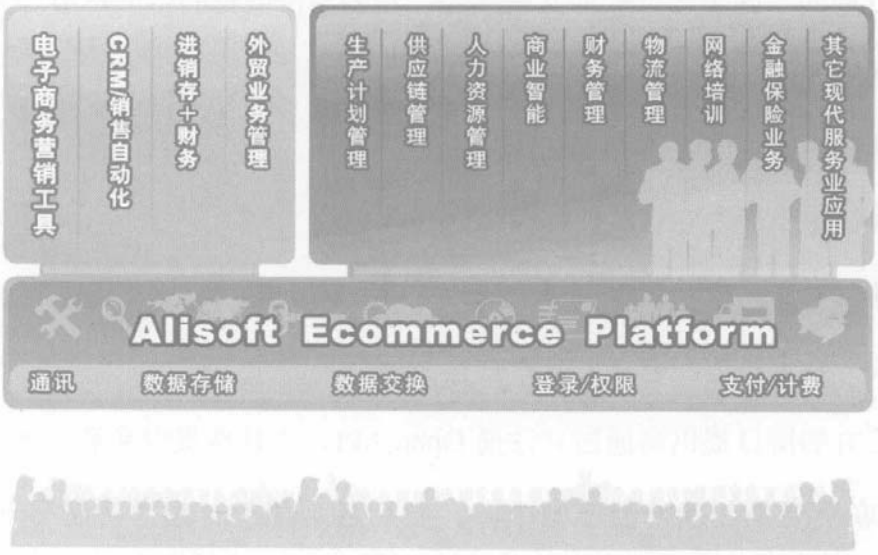


图 14-9　SaaS 软件互联

14.3　软件互联平台

SaaS 应用的互联，使得互联网应用使用形态和开发模式都有了很大的变化。应用互联过程中出现了许多新的产业链环节，这些环节一起形成了创新的软件价值链。国内外已经有很多互联网企业加入了这个价值链的建设中。图 14-10 展示的“阿里巴巴软件互联平台”生态圈，向我们揭示了开放的 SaaS 软件互联带来的革命性创新。

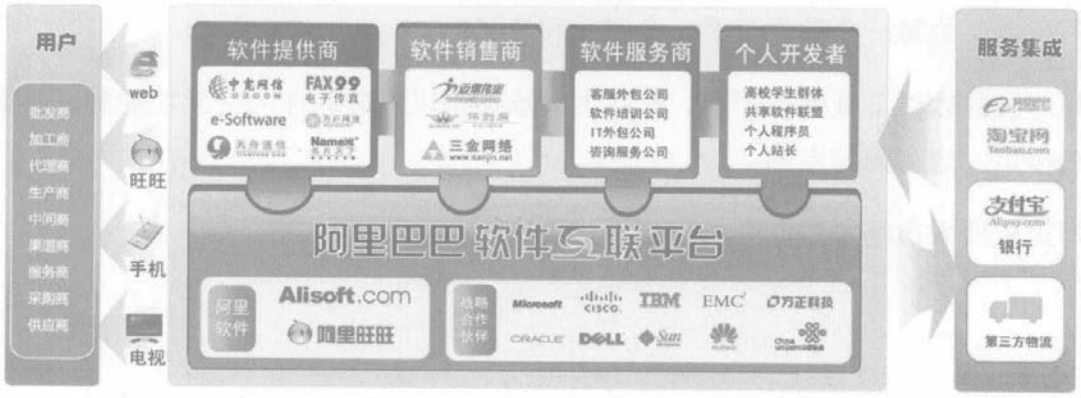


图 14-10　阿里巴巴软件互联平台

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

图 14-10 中的“阿里巴巴软件互联平台”是多条价值链的中心环节。它为：

- 最终用户带来了可选购的 SaaS 软件；
- 软件提供商（ISV）向它输入 SaaS 软件；
- 软件销售商通过它销售软件；
- 软件服务商为软件提供商提供软件支持服务；
- 个人开发者可以在它上面发布软件实现个人创业；
- 第三方的接口提供商通过它注册 Open API，供软件提供商和个人开发者使用。

软件互联平台是创新价值链的中心，它和各个角色一起搭建起了开放、协作的生态圈。“阿里巴巴软件互联平台”作为中心环节，还实现了部分 PaaS 提供商角色的功能。

14.3.1 软件大市场

软件大市场是一种软件在线销售的环境。最终用户可以在这个平台上搜索软件提供商开发的软件，订购使用这些软件，完成支付、结算、退订、续订，以及与软件开发者在线交流，提出新需求。

在软件大市场的账号体系中，一站式的、支持单点登录的账号是非常受用户欢迎的。最新用户只需要使用一个统一的账号登录软件大市场，选购软件，然后就可以启动软件。用户不再需要为每一个软件记录一个账号和密码。

与传统的软件下载网站不同，软件大市场不是以下载软件为主要功能的，而是以软件在线租用、启动为特点的。

阿里软件大市场（软件集市）界面如图 14-11 所示。

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。



图 14-11　阿里软件大市场（软件集市）界面

14.3.2　软件生态圈

软件生态圈中存在丰富的产业链角色，这些角色的合作形成软件互联的创新生态

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

圈。下面将分别介绍 ISV、ISP、PaaS 提供商等角色。

1. ISV（独立软件提供商）

ISV 在开放平台上有两种角色：

（1）应用软件的提供者和受益者

ISV 开发的软件在开放 SaaS 平台上集成后，终端用户在线使用这些应用，并可根据周期、流量、次数等方式计费结算。ISV 在这种情况下，是应用的提供者和受益者。

（2）ISP 服务的调用者和消费者

当 ISV 开发的应用需要集成 Open API，比如：网店应用需要集成淘宝的宝贝搜索、修改宝贝信息等 API 时，ISV 就是服务调用者。如果 API 的调用需要收费，则 ISV 还是这些 API 的消费者。

2. ISP（互联网服务提供商）

ISP 作为 Open API 的提供方，向 ISV 提供数据或者商业逻辑服务的开发、发布、维护，经过开放 SaaS 平台的路由和授权，使得 ISV 能够方便地接入和使用这些数据 API 服务，从而打造出面向 SCA 的应用或产品。

3. PaaS 平台运营商

通过使用 PaaS 提供商的服务，ISV 不需要购买硬件和软件，只需要简单地在 PaaS 平台上创建一个应用，得到平台颁发的 AppID 和 SecretCode，在应用开发中，使用这些标识 ISV 身份的信息，即可调用 PaaS 提供的服务。通常这只需要简单的几步鼠标点击和编码就能完成。利用 PaaS，ISV 能够创建、测试和部署一些非常有用的应用和服务，这与在基于数据中心的平台上进行软件开发相比，费用要低得多。这就是 PaaS 的价值所在。部署一个定制的数据库应用在 Force.com 上的开发时间仅仅是传统 App Server 的 1/40。PaaS 体现了互联网低成本、高效率和规模化应用的特性。

随着 PaaS 的发展，会诞生越来越多的基于互联网服务的基础设施，将大大降低技

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

术门槛，让应用更容易生成，间接带动更多的商业模式创新。尤其是资金花在软件和硬件上的比例会降低，给初创公司带来更大的生存空间。种类丰富的平台服务架构在现有的 PaaS 上，也会促成生态链的形成。最后，如果有统一的规范约束服务调用，在同样环境下，就能解决技术整合的问题。阿里软件互联平台正在从事 PaaS 服务提供商的业务，并且这个 PaaS 服务可以从 3 个层次同步发展。

（1）Open Service

开放包含了很多服务能力，其中许多是通过 Open API 实现的。从相对简单的书签服务 del.icio.us 到功能更复杂如 Amazon S3 的完全虚拟化的数据存储，通过网络服务级 API 实现的功能让人们感到惊喜。

（2）Open Data

这是一类比服务级粒度更低的 Open API，一般这类 API 提供了数据获取的能力。对于接口输出的数据一般是没有经过加工的，或只经过简单的格式化处理，使用这些接口的开发者需要自己组织完整的业务逻辑，而且需要对接口数据按实际需要再进行一些格式转换工作。虽然这些工作对 ISV 来说需要更多的后续工作，但 ISV 同时获取了最大的自由度，可以自由地对这些元始数据执行各类操作。

Open Data 对于接口服务提供商来说，技术上要求比较低，但是对于数据安全的需求较高。接口服务提供商需要自行保证开放出来的数据的稳私性。

（3）Open UI

Facebook 提供的 FBML 和校内网的 XNML 都是为了解决统一的用户体验问题。使用这类标记语言后，可以生成一些样式统一、操作方式一致的网页界面元素。而且，这些标记语言中有一部分与系统 Open Data 或 Open Service 相关联，不但可以生成界面的通用 HTML 元素，还可以自动填充系统内置的一些数据实体。

FBML 这类的 UI 语法可以算是 Open UI 的最初级形态。在使用这些标记语言的时候，开发者需要学习这些语法，这会增加一定的开发成本。另外，由于没有可视化的界面开发工具，这些界面生成的效果需要在线调试，以确定实际的显示位置、大小、内部

SaaS 架构设计
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

数据的效果等。

4. 服务接口集成商

开放的 SaaS 平台需要为 ISP 和 ISV 之间的接口交互提升便利性。这项工作的复杂性在于不同接口的参数规则化、数据消息体的格式化、接口调用的信息安全、接口的性能效率，以及接口的审计与计费工作。提供一个有效的、易用的服务接口集成系统对于 ISP 和 ISV 来说都相当重要，这关系到应用互联的用户体验。阿里软件互联平台的 SIP 系统就是一个服务接口集成系统，可以为不同的接口提供不同等级的安全检查和审计。

服务接口集成商是软件互联平台生态圈中承上启下的一个角色，它将以技术服务提供商性质的角色承担起大量的技术服务，直接面对 ISP 和 ISV 的各类需求。服务接口集成商通过为 ISP 和 ISV 的技术服务获取技术服务费。服务接口集成商在系统建设的初期需要吸引 ISP 加盟，并为 ISV 提供免费服务。随着服务接口集成商的影响力和使用数增加，ISV 将为使用 ISP 的接口付费，而服务接口集成商可以从中收获一定的技术服务费。费用一般可以按接口调用的次数、接口的流量等进行计算。

5. 软件销售商

传统软件销售商一般通过实体专卖店、连锁软件超市、电话直销、上门直销或互联网在线方式销售软件。在软件互联平台的生态圈里，这些软件销售方式同样存在，其中的互联网在线方式销售软件的模式被极大地加强，最终用户将被鼓励以在线方式购买软件，使用在线系统的统一账号登录，购买后直接启动软件。

考虑到最终用户对互联网和软件的熟悉程度不同，对 SaaS 软件的接受程度也不同，所以软件销售商还承担着与最终用户面对面沟通，解决客户疑问，引导客户快速接受 SaaS 软件的职责。由于互联网软件，特别是 SaaS 软件一般都是需要在互联网上完成用户身份验证的，所以最终用户从软件销售商手上买到的软件一般包含一对账号和密码，而不再是一张光盘。因此软件销售商的销售技巧和销售方式都与传统的软件销售不同。在这个生态圈刚刚建立的初期，软件销售商的角色是非常必要的。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

6. 其他服务商

除了以上提到的生态圈角色以外，软件互联平台中还存在很多其他类型的软件服务商。

由于软件互联平台的软件提供商比传统软件公司更容易获得外地客户，这些外地客户对于软件的学习使用、建议需求问题的解决都有别于传统软件公司。

因此，专业的软件使用**培训支持服务提供商**、软件内的二次定制开发服务提供商将在客户所在地进行服务，而软件提供商则专注于软件功能和质量的提升。

SaaS 软件开发需要处理很多应用互联的技术特性，因此这些提供这类开发支持的**开发工具服务商**也是重要的一个角色。传统软件的开发工具一般针对单机版软件、单一 C/S 软件或独立 Web 系统提供服务；而 SaaS 软件的开发要求实现方便的二次开发和界面功能的自定义。Salesforce 提供的在线软件开发工具就是一个 SaaS 在线功能包定制的工具。SaaS 软件在与其他应用互联时需要调试远程的接口参数和异常信息，现在的软件开发工具可以部分支持这些功能，但还有很大的改进空间。

SaaS 软件需要账号登录后才可以使用，因此，一些**独立的 Passport 服务提供商**已经逐渐形成。Google Accounts 和 OpenID 的开放账户系统已经在互联网网站抢占了一些占有率，一些 SaaS 软件已经开始同时支持本地账号和这些开放的账户接口。

开放的 SaaS 软件互联平台已经开始了新的产业链建设。在这个产业链的建设过程中，新的软件生态圈也将慢慢形成。在这个过程中还会有很多新的角色出现或消失，并且还会有些角色同时承担多个服务提供商的职能。

14.4 小结

本章介绍了开放 SaaS 平台的概念。通过对 PaaS 几种商业模式的介绍，使我们对开放平台的真实特性更加清晰。我们需要的不仅仅是狭义平台，更应该创造条件，完善价值链，吸引更多的角色参与到平台建设中，共同打造软件大市场和软件生态圈。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

第15章 结束语

15.1 SaaS 发展趋势……325
15.1.1 SaaS 平台化……325
15.1.2 SaaS 产业链……326
15.1.3 SaaS 移动化……327
15.1.4 服务即软件……328
15.2 软件行业的未来……330
15.2.1 通过互联网使用软件的用户越来越多……330
15.2.2 软件和服务的界限会越来越模糊……330
15.2.3 软件和服务会越来越个性化……331
15.2.4 软件产业大融合……331

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

人类社会进入 21 世纪之后，作为现代社会的支柱产业——软件产业的发展势头突飞猛进。特别是以互联网为核心的在线软件服务（SaaS）的发展，将在国内外软件市场掀起一场势不可当的新技术革命。众多传统软件厂商开始转向在线软件的研发大潮，大量外资企业抢滩和布局中国的新兴市场，越来越多的风险投资投向大大小小的在线应用项目，整个软件产业向 SaaS 转型已成必然趋势。

从 SaaS 产业的发展趋势来看，我们可以预测整个软件产业的发展方向。只要我们能清晰地看到这个发展方向，就能抓住软件的未来，从而开创软件产业新的明天。

15.1 SaaS 发展趋势

传统软件走向 SaaS 软件，是互联网发展到一定阶段的必然产物。网络覆盖的广泛性，网络带宽的不断提升，互联网一系列量变因素积累到一定程度就产生了质变，SaaS 也因此而诞生。但互联网还在飞速发展，SaaS 软件最终又会发展成什么样子？SaaS 将要走向何方？这都是研究 SaaS 需要思考的问题。

15.1.1 SaaS 平台化

从 SaaS 商业模式来看，SaaS 软件盈利的关键是服务的低成本和用户数量之间的平衡。低成本是 SaaS 软件的最大优势，但光有低成本是不行的。即使一个 SaaS 应用的收费很低，但未必能引来大量的用户群。如果低成本的 SaaS 应用没有大量用户群的支持，肯定会导致商业上的失败。

于是，一些拥有大量在线客户群的企业看到了这个商机，纷纷打造 SaaS 运营平台，将自己掌握的庞大客户资源开放给第三方软件厂商。此举既能让第三方软件厂商迅速找到所需要的在线客户群，又能为这些客户群提供无限丰富的 SaaS 应用，而自己也能从中分得一杯羹，真可谓是一举三得。

所谓 SaaS 运营平台指的是可以支撑多种 SaaS 应用的，集接入、运行、推广、销售、计量、支付和维护为一身的综合环境。也就是说，任何第三方软件上开发的 SaaS 应用

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

只要接入到 SaaS 运营平台，就能立即投入运行并找到买家和收到服务费。

从另一方面讲，SaaS 应用需要面对大量客户，这要求承载 SaaS 应用的后台环境必须足够健壮、稳定和安全。这不但要求 SaaS 应用提供商搭建支撑巨量访问的硬件环境，还需要提供可靠的实时备份和异地灾备设施，以及安全的网络环境。因此，SaaS 后台的投入是非常巨大的，其成本非一般的小企业可以承受。

而有实力的大企业可以投入大量资金集中打造 SaaS 运营平台，提供全方位的 SaaS 应用托管服务，既能保证 SaaS 应用的高效和可靠运行，又能提供各种安全的备份设施，充分保障 SaaS 运行所需要的环境。从整体上说，这样的集成 SaaS 运营平台的总体成本是非常低廉的，保证了大量的第三方软件低成本接入，从而使得第三方软件商有能力为广大用户提供低成本的 SaaS 应用。这也是市场催生 SaaS 平台的第二个原因。

目前，一些有实力的大企业开始打造 SaaS 运营平台。典型的有 Salesforce 的 Force.com 平台、Google 的 App Engine 平台，以及阿里软件的 AEP 平台。这些平台各有优缺点，关注的重点内容也不尽相同，但有一个共同点就是为广大的第三方软件商提供一个 SaaS 软件的接入和运营环境。随着 SaaS 平台技术的不断发展，SaaS 运营平台必将成为支持整个 SaaS 产业的关键基础设施。

15.1.2 SaaS 产业链

SaaS 软件发展的一个重要趋势就是形成 SaaS 产业链。所谓产业链，围绕相同的经济价值，各类型企业参与，自然形成紧密关联的供需逻辑结构。当大量企业参与 SaaS 软件的供需环节，并形成一定量的规模经济效益时，我们就可以说形成了 SaaS 产业链。

SaaS 产业链的供需关联关系是以 SaaS 运营平台为纽带的。互联网上的所有上网客户都是 SaaS 产业的潜在需求方，客户旺盛的需求吸引大量的软件商开发 SaaS 应用。然而，客户不可能一家家地去寻找 SaaS 软件厂商的应用，而 SaaS 软件商也不可能一家家地去推销自己的应用。于是，SaaS 产业必然出现 SaaS 平台运营商，来填补 SaaS 产业空缺的枢纽环节。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

SaaS 产业链的基本形态就是，SaaS 软件商将 SaaS 应用放到 SaaS 运营商的平台上，而广大用户到 SaaS 运营平台上寻找自己所需的应用服务，SaaS 运营平台成为 SaaS 应用的超级市场。在 SaaS 产业链，第三方 SaaS 软件商是服务的最终生产商，SaaS 运营商是服务的推广和销售渠道，广大互联网用户是 SaaS 应用的最终消费者。SaaS 产业链上各类企业的供需关系清晰明确，各自的职责也非常分明，也便于政府实施产业监管。

在 SaaS 产业链中，SaaS 平台运营商起着至关重要的纽带作用。SaaS 平台运营商需要搭建一个 SaaS 软件的运营平台，进行市场推广，吸引广大客户来 SaaS 平台订购和使用软件。对于用户来说，到一个集中的地方订购和使用所需要的软件应用，比零零散散地到处寻找和采购要方便很多。这就好比人们更愿意去超市购物，而不愿意自己到处找供应商一样，主要是因为方便。对于广大第三方软件商来说，利用 SaaS 平台运营商的软件平台，可以直接获得一大群客户，省去了自己做市场推广的麻烦。因此，大力发展以 SaaS 运营平台为基础的 SaaS 产业链，是一个必然的趋势。

作为 SaaS 产业链的基础，SaaS 运营平台需要提供许多必要的基础服务。这些基础服务主要包括数据存储服务、应用托管服务、用户身份认证、网络安全服务、计量收费服务和统一的技术支持服务。而这些基础服务在基础设施和软硬件环境上都有较高的要求，同时还需要运营平台商有很强的市场推广和营销能力。因此，对于 SaaS 软件运营平台的建设，需要巨大的投入和强有力的市场推广能力，一般的小型软件厂商没有这样的实力，只有具有强大经济实力和营销能力的大型软件公司才适合运营 SaaS 软件平台。

事实上，阿里软件互联平台已经逐渐形成了 SaaS 产业链的雏形，众多的第三方软件商也在这个产业链中取得了可观的规模效益，广大用户也在 SaaS 应用的选择上获得了实质的方便和成效。由此可见，SaaS 产业链理论已经初显成效，将为整个软件行业的转型起到示范和推进作用。

15.1.3 SaaS 移动化

可以毫不夸张地说，未来的“手机”定义将不再是“手持电话机”，而是“手持计算机”。随着智能手机的普及，手机不再仅仅是电话加上一些 MP3 或游戏的智能玩具。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

事实上，现在的手机已经具备了相当于初级桌面电脑的计算能力，越来越多的应用软件被移植到手机上。这些都让手机日益成为个人用户最重要的信息终端。

SaaS 应用作为一个无须安装即可使用的方便服务，也同样适合手机环境。因此，一些企业看到了手机应用的 SaaS 商机，专门开发针对手机的 SaaS 应用。目前，尽管手机屏幕还很小，展示的信息有限，操作起来没有桌面电脑方便，所能提供的应用服务也有限。但是，手机 SaaS 应用的趋势却不可阻挡。

SaaS 模式提倡随时随地可以使用软件的思想，而 SaaS 的移动化正是这种思想的最佳体现。毕竟台式电脑无法携带，笔记本电脑似乎又太大太重，而手机却是小巧玲珑、随身携带的；笔记本电脑需要有网络环境，需要配置无线网卡等上网设施，而手机本身就是走移动通信网络的智能终端，随时随地都在网上。应该说没有哪种终端能像手机一样适合 SaaS 应用所需的所有特征。

手机终端还有一个特别吸引众多 SaaS 开发商的地方就是：手机 SaaS 应用可以到达企业决策者手中。对于许多企业老板或经理来说，他可能无须使用电脑，但却不能不使用手机。尽管在许多企业中，也为高层领导配备了豪华的台式电脑或笔记本电脑，但往往都是形同虚设，因为老板们根本就没时间用。这种现象在中国的中小型企业中特别普遍，而手机就成了这些决策者最好的管理和控制平台。

正是因为手机 SaaS 应用广阔而诱人的前景，使得众多 SaaS 软件商抢滩和挖掘这块沃土。继 Apple 公司的 iPhone 之后，Google 公司发布了智能手机 Android，这标志着 Google 将其产业发展战略延伸到了通信行业。Android 手机不但提供了丰富的移动商务应用，更重要的是它同时也是一个强大的手机 SaaS 软件开发和运行的平台。Google 大力发展手机战略，无疑让 Google 在手机 SaaS 应用的竞争中占据了领导地位。

15.1.4 服务即软件

随着 SaaS 软件产业的发展，人们在 SaaS 软件的实践应用中不断地发现和总结整个产业的规律。一些新的思想和理论又开始成熟和发展，其中就有“服务即软件”的理论思想。

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

“服务即软件”的思想最早是由阿里软件提出来的。阿里软件在研究了 SaaS 软件互联平台的各种统计数据和客户调研结果后发现，最受用户欢迎的 SaaS 应用并不是那些功能强大或者价格低廉的应用项目，而是那些功能简单但方便快捷的 SaaS 应用。这些受欢迎的 SaaS 应用的背后，往往是基于其他系统提供的增值服务，或者干脆就是传统服务的信息化形式而已。

这些第三方系统提供的增值服务，或者有传统服务支撑的 SaaS 应用服务，才是最终用户最需要的服务。只是因为 SaaS 软件让这些服务使用起来更加方便和快捷，用户只需利用 SaaS 运营平台，足不出户即可获得所需要的服务结果。由此可见，用户真正需要的是各种各样的实用服务，SaaS 软件只是提供这些服务的一种先进形式。

于是，阿里软件率先提出了“服务即软件”的思想。Service as a Software 缩写同样也是 SaaS，但其重点完全放到了 Service 之上，即强调的是服务。如果说“软件即服务”是找准了软件的最终位置，则“服务即软件”才是 SaaS 的最终目的。或许“服务即软件”才是 SaaS 产业发展的终极目标，才是 SaaS 的真谛。

在“服务即软件”的理论中，SaaS 软件只是提供服务的一种手段。也就是说，服务还可以用其他的手段来提供，只是用 SaaS 软件的形式会更加方便和快捷，从而更加先进。事实上，大多数传统服务都可以通过 SaaS 软件来实现服务的信息化工作。换句话说，如果“软件即服务”强调的是软件的服务化，而“服务即软件”则是强调服务的信息化。

“服务即软件”的思想无疑给 SaaS 行业指出了一个重要的方面，SaaS 并不只是软件的服务化，更重要的是服务的软件化。也就是说，如果某项服务是广大用户所需要的，就应当以软件的方式引入 SaaS 平台以方便用户使用。这一思想无疑让整个 SaaS 行业的战略思路变得更加明确和清晰。

事实上，SaaS 技术是促进现代服务发展的有效手段，而“服务即软件”理论更加明确了 SaaS 技术就是为了服务的发展。现代服务业是指那些不生产商品和货物的产业，主要有信息、物流、金融、会计、咨询、法律服务等行业。现代服务业的发展本质上来自于社会进步、经济发展、社会分工的专业化等需求，具有智力要素密集度高、产出附

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

加值高、资源消耗少、环境污染少等特点。现代服务业既包括新兴服务业，也包括对传统服务业的技术改造和升级，其本质是实现服务业的现代化。

“服务即软件”的 SaaS 理论为发展现代服务业打下了坚实的理论基础。

15.2 软件行业的未来

15.2.1 通过互联网使用软件的用户越来越多

未来的网络是无处不在，人们随时随地都身处网络之中，根本就不存在“上网”的概念。因为，与人们工作和生活息息相关的各种设备都是实时连入互联网的，电脑、手机、电视、冰箱、微波炉甚至汽车等，都是网络的终端。几乎任何软件都会自然而然地与网络关联起来，可以说软件都是网络化的软件。

而整个软件行业也必然向着这个方向发展，以适应越来越多的在线网络软件用户的需求。其实，这也就是云计算发展的终极目标。未来的软件都是云计算上的软件，软件开发在云端，运行也在云端，互联网只是连接人们和云端的基础设施。

互联网的未来就是软件业的未来。现在，使用互联网的人越来越多，特别是中国互联网用户已经成为全球最大的用户群。通过互联网使用软件的用户也就越来越多，这为软件行业的未来带来无限的希望。

15.2.2 软件和服务的界限会越来越模糊

SaaS 的概念首次将软件和服务联系在一起，从而改变了人们对软件的看法。但这仅仅是一个开始，人们会逐渐发现越来越多的事情可以通过互联网来完成。

在互联网上可用服务日益增多的同时，互联网上的软件却变得越来越不重要了。互联网中的一种服务可能有多家软件商提供产品支持。甚至，由于服务的规范化，支持同一服务的软件都必然采用相同的用户界面和操作模式。因此，人们在消费某项服务时可以自由选择喜欢的软件，或者干脆就不关心用什么软件。就像打电话一样，人们只关心

互联网时代的软件革命
Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

通信服务怎样，而越来越不关心用什么手机了。

软件会演变成提供服务的工具，或者说软件和服务的界限将越来越模糊，是未来软件发展的必然趋势。从“软件即服务”到“服务即软件”，整个软件产业正大踏步地朝着向现代服务业融合的方向前进。

15.2.3 软件和服务会越来越个性化

当基本的需求满足之后，人们总会追求个性化的精神需求。毕竟，大多数人都想与众不同嘛。同样，当人们已经习惯了已有软件和服务之后，就会提出对软件和服务的个性化需求。因此，未来的软件一方面要为用户提供标准的基础服务，另一方面需要能为用户提供个性化的软件和服务。

软件和服务的个性化主要体现在许多方面。比如，服务层次个性化：同样的服务可能会分档次提供，有的喜欢简单的低档服务，有的喜欢中档，而有的喜欢豪华高档服务。再如，软件服务的用户喜好个性化：界面的颜色、操作模式、体验的风格，甚至虚拟服务场景等，都会有个性化需求。总之，个性化的需求总会在工作和生活中的各个方面产生。

因此，作为具有超前眼光的软件开发商，应当看到软件和服务的个性化发展趋势。积极主动地设计和开发适合个性化需求的软件，为未来软件的发展铺平道路。

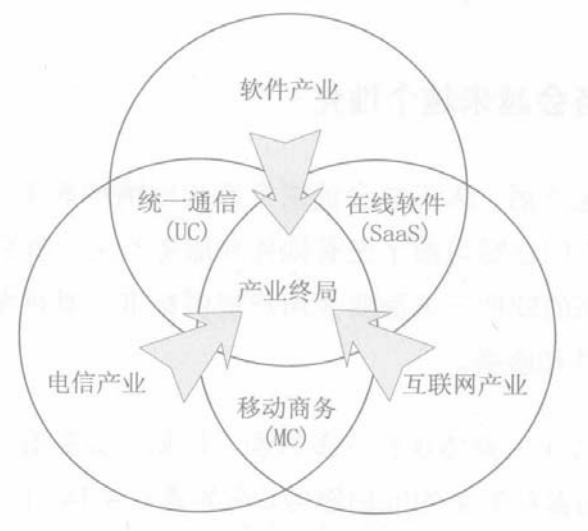
15.2.4 软件产业大融合

要预测软件产业的发展趋势，必须站在一个更宏观的战略高度来看问题，同时还必须了解其他产业的发展趋势。

事实上，软件产业、互联网产业和电信产业是现代信息社会的三大产业支柱。这三大支柱产业又并非各自独立发展的，它们之间总是在相互依赖、相互影响和相互促进中协同发展。而进入 21 世纪之后，软件产业、互联网产业和电信产业更是出现了相互融合的发展趋势。

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

由于各产业内部需求的压力，使得软件产业、互联网产业和电信产业之间不断地相互挤压和渗透。于是，互联网产业与软件产业的融合产生了在线软件（SaaS）；电信产业与软件产业的融合产生了统一通信（UC）；互联网产业与电信产业的融合产生了移动商务（MC）。三大产业的关系和发展趋势如下图所示。



我们可以推断，软件产业、互联网产业和电信产业相互挤压和渗透的最终结果，必然是三大产业的彻底融合。这就是软件、互联网和电信产业发展的产业终局，这就是软件产业的未来！

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

参 考 资 料

- Architecture Strategies for Catching the Long Tail

 http://msdn.microsoft.com/en-us/library/aa479069.aspx

- 运用 RUP 4+1 视图方法进行软件架构设计

 http://www.ibm.co/developerworks/cn/rational/06/r-wenyu/index.html;

- 架构蓝图——软件架构“4+”视图模型

 http://www.ibm.com/developerworks/cn/rational/r-4p1-view/;

- Model Driven Architecture

 http://www.omg.org/mda/;

- Multi-Tenant Data Architecture:

 http://msdn.microsoft.com/en-us/library/aa479086.aspx;

- Yahoo!团队实践分享——网站性能优化的 34 条黄金守则

 http://developer.yahoo.com/performance/rules.html

- HTTPS 传输协议原理介绍，Hatter Jiang Apr 9th, 2008

- HTTPS 原理

 http://wireless.javaeye.com/blog/248473

- Google Gears

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

http://gears.google.com/

- 维基百科

 http://zh.wikipedia.org/wiki/SQLite

- GFS

 http://labs.google.com/papers/gfs.html

- MapReduce

 http://research.google.com/archive/mapreduce.html

 Distributed Programming with MapReduce, Jeffrey Dean,Sanjay Ghemawat, *Beautiful Code*, 2007, Chapter 23.

- Hadoop

 http://hadoop.apache.org/core/

- PGPool-II

 http://pgpool.projects.postgresql.org/

- MySQL 5.0 Reference Manual

 http://dev.mysql.com/doc/refman/5.0/en/index.html

- PostgreSQL 8.3.3 Documentation

 http://www.postgresql.org/docs/8.3/interactive/index.html

- Consistent hashing

 http://en.wikipedia.org/wiki/Consistent_hashing

互联网时代的软件革命

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

- iBATIS for Java

 http://ibatis.apache.org/

- CSDN 关注 OpenAPI

 http://subject.csdn.net/Programer_0807_OpenAPI/

- 阿里软件开发者博客

 http://www.alisdn.com/

- 叶军的博客

 http://www.yejun.cn/

- 文初的专栏

 http://blog.csdn.net/cenwenchu79

- 阿里巴巴软件互联平台技术白皮书
- Alisoft Service Integration Platform 技术白皮书
- 阿里软件服务集成平台 REST Style Service ISV 接入规范
- 淘宝 API 参考手册 V1.0

Java学习群：72030155
好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间；可以联系群主获取更多大型企业内部技术教程。

Java学习群

72030155

进群可以免费获取视频教程以及每日免费听老师讲课